Федеральное государственное бюджетное образовательное учреждение
высшего профессионального образования
«Южно-Уральский государственный университет»
(национальный исследовательский университет)
Факультет «Математики, механики и компьютерных наук»
Кафедра «Вычислительная механика сплошных сред»

ДОПУСТИТЬ К ЗАЩИТЕ
Заведующий кафедрой ВМСС

д. ф.-м. н., профессор
Ковалев Ю.М.
_____ 2016 г.

Численное исследование влияния экранирующих слоев газовзвеси на отражение ударных волн от жесткой стенки

ЮУрГУ – 010800.62-2016.661.ВКР

Руководитель работы
к., ф.-м. н., доцент кафедры ВМСС
_____/Н.Л. Клиначева
«3» июня 2016 г.

Автор работы
студент группы ММиКН-470
_____/П.Е. Беляев.
«3» июня 2016 г.

Нормоконтролер
Старший преподаватель кафедры ВМСС
_____/Е.Е. Пигасов
«3» июня 2016 г.

Челябинск 2016

Министерство образования и науки Российской Федерации
Федеральное государственное бюджетное образовательное учреждение
высшего профессионального образования
«Южно-Уральский государственный университет»
(национальный исследовательский университет)
Факультет «Математики, механики и компьютерных наук»
Кафедра «Вычислительная механика сплошных сред»

УТВЕРЖДАЮ
Заведующий кафедрой,
д. ф.-м. н., профессор
_____/Ю.М. Ковалев
10 февраля 2016 г.

ЗАДАНИЕ

студенту группы ММиКН-470

Беляеву Павлу Евгеньевичу

на выполнение выпускной квалификационной работы

по направлению 010800.62.–Механика и математическое моделирование.

1. **Тема выпускной квалификационной работы** «Численное исследование влияния экранирующих слоев газовзвеси на отражение ударных волн от жесткой стенки»

(Утверждена приказом ректора от «15» апреля 2016 № 661)

2. **Перечень подлежащих исследованию вопросов**
   - 2.1. Обзор литературы по решению задачи о распространении плоской ударной волны в воздухе, взаимодействии с гетерогенным слоем
   - 2.2. Составление программы расчёта.
   - 2.3. Решение задачи для заданных начальных условий.
3. **Календарный план подготовки выпускной квалификационной работы**

Наименование этапов выпускной квалификационной работы	Срок выполнения этапов работы	Отметка о выполнении
1.Обзор литературы	10.02.16-10.03.16	
2. Изучение численного метода для решения поставленной задачи. Решение тестовой задачи.	11.03.16-01.04.16	
3.Получение результатов, формулировка выводов, структурирование текста	01.04.16-15.05.16	
4.Подготовка текста	16.05.16-01.06.16	
5. Проверка и рецензирование работы руководителем, исправление замечаний. Подготовка доклада и текста выступления. Внешнее рецензирование.	02.06.16-13.06.16	
6. Защита выпускной квалификационной работы.	17.06.16	

4. Дата выдачи задания 10 февраля 2016 г.

Руководитель работы

к.ф.-м.н., доцент кафедры ВМСС _____/ Н.Л. Клиначева

Задание принял к исполнению_____/ П.Е. Беляев

ОГЛАВЛЕНИЕ

ОГЛАВЛЕНИЕ 5

ВВЕДЕНИЕ 6

1. ОБЗОР ЛИТЕРАТУРЫ 7

2. ПОСТАНОВКА ЗАДАЧИ 11

2.1. Начальные условия 12

2.2. Граничные условия 12

3. ЧИСЛЕННЫЙ МЕТОД 13

3.1. Основные сведения 13

3.2. Предварительные преобразования 15

3.3. Эйлеров этап. 16

3.4. Лагранжев этап 18

3.5. Заключительный этап 19

4. РЕЗУЛЬТАТЫ 21

ЗАКЛЮЧЕНИЕ 26

ПРИЛОЖЕНИЕ 1 27

ПРИЛОЖЕНИЕ 2 34

ЛИТЕРАТУРА. .....

ВВЕДЕНИЕ

Актуальность работы:

Взрывы, аварийные разрывы ёмкостей или газопроводов высокого давления, а также полёты в атмосфере со скоростями, превышающими скорость звука приводят к образованию ударных волн, течений газа со скачкообразным изменением параметров среды, которые, в свою очередь, могут представлять опасность для инженерных сооружений и персонала. Данный факт демонстрирует необходимость в разработке средств защиты от ударных волн.

Цель работы:

Изучить влияние экранирующего слоя газовзвеси на отражение нестационарной ударной волны от неподвижной стенки.

Задачи:

1. Изучить математическую модель двухскоростного двухтемпературного континуума [2].
2. Построить численный алгоритм на основе модели [2], позволяющий смоделировать взаимодействие ударной волны со слоем газовзвеси и отражение ударной волны от стенки.
3. Сравнить полученные результаты с результатами, приведёнными в [3].

В данной работе представлены результаты численного исследование влияния экранирующего слоя газовзвеси на отражение плоской нестационарной ударной волны от жёсткой стенки.

1. ОБЗОР ЛИТЕРАТУРЫ

В [1] рассмотрены нестационарные, в том числе волновые, вибрационные и фильтрационные, а также стационарные движения различных гетерогенных, или многофазных сред, широко представленные в различных природных процессах и областях человеческой деятельности.

Губайдуллин А. А., Ивандаев А. И. и Нигматулин Р. И. в [2] описывают математическую модель двухскоростного двухтемпературного континуума и модификацию метода крупных частиц, позволяющую использовать его для расчёта нестационарных течений многофазных сред, на примере одномерного нестационарного течения двухфазной смеси. Также в статье описан алгоритм локализации контактного разрыва в несущей фазе, позволяющий добиться точного расчёта границ этого разрыва.

В работе [3] Ивандаев А. И. и Кутушев А. Г., проводя численный эксперимент, рассматривают влияние экранирующего слоя газовзвеси на набегающую на стенку нестационарную ударную волну. Рассматривается следующая задача: на однородный слой газовзвеси набегает плоская ударная волна конечной длительности. Движение среды рассматривается при основных допущениях [1] с предположением, что расстояние, на котором меняются параметры среды много больше, чем размеры частиц или расстояния между ними, частицы имеют сферическую форму и одинаковый размер, эффекты теплопроводности и вязкости учитываются только в процессах межфазного взаимодействия, не происходит дробление и столкновение частиц. Неустановившееся течение газовзвеси описывается моделью двухскоростного двухтемпературного континуума [1]. Несущий континуум считается идеальным калорическим газом, а вещество дисперсной фазы – несжимаемым. Расчёты проводились с помощью метода крупных частиц с выделением различных контактных границ при помощи алгоритмов локализации типа [2]. Авторами работы показано, что

наличие экранирующего слоя приводит к заметной трансформации ударной волны: её размазыванию, ослаблению и удлинению. Приведено подробное описание процесса отражения ударной волны от неподвижной стенки. Проведён анализ влияния массовой доли дисперсной фазы в экранирующем слое на характер отражения ударной волны. Проанализировано импульсное взаимодействие ударной волны со стенкой. Показано, что увеличение массовой доли частиц от 0 до 8 вызывает снижение максимального давления на стенке примерно в два раза.

В книге [4] даётся подробное описание трёх численных методов: метод частиц в ячейках, метод свободных точек и метод крупных частиц. Данные методы позволяют производить численное интегрирование системы уравнений газовой динамики. Особое внимание уделено методу крупных частиц.

В статье [5] Ковалёвым Ю. М. и Пигасовым Е. Е. приводится модификация метода крупных частиц, позволяющая использовать его с моделью [2]. Модификация заключается в использовании на этапе Эйлера неявной аппроксимации в расчёт членов, описывающих межфазное взаимодействие, и в расчёт промежуточных скоростей. Введение модификации увеличивает стабильность вычислений и позволяет производить расчёт при больших числах куранта, чем оригинальный метод.

Монография [6] носит обзорный характер. В ней систематизируются данные по исследованию дисперсных потоков с теплообменом и изотермических. Автором рассматриваются наиболее простые уравнения движения среды, затем устанавливается влияние таких факторов, как наличие частиц, их размеров, концентрации и т.п. Показывается недостаточность исследований в данной области и приводится методика более правильной, по мнения автора, оценки данных явлений.

В работе [7] Кругликов Б. С. и Кутушев А. Г. рассматривают проблему ослабления воздушных ударных волн при помощи защитных

экранов. Изучается возможность использования модели замороженной газовзвеси для описания законов затухания и взаимодействия с преградами ударных волн в слоях запыленного газа и решёток. Приведены результаты исследований для коротких ударных волн, когда времена релаксации скоростей и температур значительно больше длительности фазы сжатия. Газ с решётками моделируется в рамках модели двухтемпературного континуума с неподвижной конденсированной фазой. Узлы решётки рассматриваются как частицы замороженной взвеси. Влиянием связей решёток пренебрегают в силу допущения о малости их по сравнению с узлами. Расчёты проводились для смеси воздуха с частицами железа в широком диапазоне объёмных содержаний частиц ($0 \leq \alpha_2 \leq 0.065$) и их диаметров ($60$ мкм $\leq d \leq 1.2$ мм) в различных комбинациях. В ходе работы показано, что для оценки эффективности ослабления ударных волн посредством экранирующих слоёв запыленного газа использование модели замороженной газовзвеси вполне допустимо, а наличие экрнирующих слоёв мелкоячеистых решёток даже с небольшим объёмным содержанием приводит к сильному ослаблению воздушных ударных волн. Результаты работы могут использоваться при анализе распространения ударных волн конечной длительности и умеренной интенсивности в запыленных газах.

В работе [8] приводятся результаты численного исследования течения газа в плоских ударных волнах, проходящих над выемкой прямоугольной формы, в которой находится газовзвесь. Для моделирования движения газа с частицами используется модель многофазной сплошной среды. Расчёт проводится для нестационарной ударной волны, интенсивность которой характеризуется числом маха $M = 4.2$, начальной длиной 0.45 м, проходящей над выемкой глубиной и шириной 0.13 м. Газовзвесь в выемке состоит из воздуха и частиц графита с массовой долей $\rho_{20}/\rho_{10} = 1$ и диаметром $d = 60$ мкм. Результаты исследования показывают, что при прохождении ударной волны над

выемкой с запыленной областью газ в выемке сильно сжимается, а затем, когда фронт волны уходит далеко, давление падает и газовзвесь попадает из выемки наружу, при этом образуются вихревые течения.

Гришин Ю.А. и Зенкин В. А. в [9] приводят модификацию метода крупных частиц, позволяющую проводить расчёт при числах Куранта близких и даже превышающих единицу. Предлагаемая модификация заключается в использовании в расчётах промежуточных значений давления, полученного из разностного представления преобразованного уравнения сохранения энергии.

В статье [10] Ковалёв Ю. М. приводит результаты численного исследования влияния неоднородностей в экранирующих слоях решёток на ослабление и отражение ударных волн. Расчёты показывают, что в разрывах экранирующего массива, в следствии отражения от границ разрыва, образуются вторичные ударные волны. Учёт данного факта позволяет уточнить моделирование прохождения ударных волн через лесные массивы. Результаты численного моделирования прохождения ударной волны через лесные массивы, в котором лесной массив представлен неоднородными экранирующими решётками значительно лучше согласуется с результатами экспериментов, чем при представлении лесного массива однородными.

2. ПОСТАНОВКА ЗАДАЧИ

Пусть на однородный слой газовзвеси ширины l_s, экранирующий жесткую неподвижную стенку, набегает плоская ударная волна конечной длительности. Требуется изучить влияние слоя на воздействие волны на преграду. Движение смеси газа с твердыми дисперсными частицами будем рассматривать при основных допущениях [1-3]. Предположим, что расстояния, на которых средние параметры меняются, значительно больше размеров частиц и расстояний между ними (вне поверхностей ударных скачков в несущей фазе), частицы сферические, смесь монодисперсная, эффекты вязкости и теплопроводности существенны лишь в процессах межфазного взаимодействия, дробление и столкновение частиц отсутствуют, фазовые превращения не происходят. Для описания неустановившихся течений газовзвесей используем модель двухскоростного и двухтемпературного континуума [1]. Несущий континуум будем считать идеальным калорически совершенным газом, а вещество дисперсной фазы — несжимаемым. Расчёты будем проводить численно интегрируя замкнутую систему дифференциальных уравнений плоского одномерного течения газовзвеси [1] методом крупных частиц [4] с применением модификации [5] на этапе Эйлера.

Сила межфазного взаимодействия f представляет собой совокупность нескольких сил различной природы: сила вязкого трения f_μ, Архимеда f_A, присоединённой массы f_m и Бассэ f_B [6]:

$$f = f_\mu + f_A + f_m + f_B$$

Однако вклад наследственной силы Бассэ в f при больших числах Рейнольдса Re_{12} относительного обтекания частиц менее значим, чем вклад сил Архимеда и присоединённой массы, вклад которых обычно значительно меньше вклада силы вязкого трения[6].

2.1. Начальные условия

Параметры газа за ударным скачком (отмечены индексом н) связаны с параметрами газа перед скачком (отмечены индексом 0) соотношениями Гюгонио

$$\frac{u_{1\text{н}}}{a_0} = \frac{2}{\gamma+1}\left(M_\text{н} - \frac{1}{M_\text{н}}\right),$$
$$\frac{p_\text{н}}{p_0} = 1 + \frac{2}{\gamma+1}\left({M_\text{н}}^2 - 1\right),$$
$$\frac{\rho^0_{1\text{н}}}{\rho^0_{10}} = \frac{(\gamma+1){M_\text{н}}^2}{2+(\gamma-1){M_\text{н}}^2}.$$

Здесь

$$\frac{u_1}{u_{1\text{н}}} = \frac{x}{x_\text{н}} \qquad \frac{\rho^0_1}{\rho^0_{1\text{н}}} = \xi \qquad \frac{p}{p_\text{н}} = \xi^\gamma \qquad \alpha_1 = 1 \qquad (0 \le x \le x_\text{н})$$

$$u_1 = 0 \qquad \frac{\rho_1}{\rho^0_{1\text{н}}} = 1 \qquad \alpha_1 = 1 \qquad \frac{p}{p_0} = 1 \qquad (x_\text{н} \le x \le x_s)$$

$$u_1 = 0 \qquad \frac{\rho_1}{\rho^0_{1\text{н}}} = \alpha_{10} \qquad \alpha_1 = \alpha_{10} \qquad \frac{p}{p_0} = 1 \qquad (x_s \le x \le x_w)$$

$$u_2 = 0 \qquad \frac{\rho_2}{\rho^0_{20}} = 1 \qquad \alpha_2 = \alpha_{20} \qquad \frac{T_2}{T_0} = 1 \qquad (x_s \le x \le x_w)$$

$$\xi = \left(1 - \frac{(\gamma-1)(u_{1\text{н}} - u_1)}{2a_{1\text{н}}}\right)^\gamma$$

2.2. Граничные условия

Численные расчёты проводились на разностной сетке, покрывающей ограниченный отрезок $[0; x_w]$ оси x. Его левая граница $x = 0$ всегда оставалась открытой, и на ней ставились условия свободного протекания фаз:

$$u_i(0_+, t) = u_i(0_-, t)\ (i = 1{,}2)$$

При $x = x_w$ (жёсткая стенка) задавались условия нулевой скорости для газа и свободного протекания для частиц (последнее моделировало неупругое столкновение частиц со стенкой):

$$u_1\left(x_{w+}, t\right) = -u_1\left(x_{w-}, t\right), \qquad u_2\left(x_{w+}, t\right) = u_2\left(x_{w-}, t\right)$$

3. ЧИСЛЕННЫЙ МЕТОД

В данной работе для получения результатов применяется метод крупных частиц с модификацией эйлерова этапа описанной в [5] и основными допущениями, описанными в [1]. Используется математическая модель, предложенная в [2].

3.1. Основные сведения

Основная идея метода крупных частиц состоит в расщеплении по физическим процессам исходной нестационарной системы уравнений Эйлера, записанной в форме законов сохранения. Среда моделируется системой из жидких (крупных) частиц, совпадающих в данный момент времени с ячейкой эйлеровой сетки. Стационарное решение задачи, если оно существует, получается в результате установления, поэтому весь процесс вычислений состоит из многократного повторения шагов по времени. Расчет каждого временного шага (вычислительного цикла) в свою очередь разбивается на три этапа:

I. Эйлеров этап: пренебрегаем всеми эффектами, связанными с перемещением элементарной ячейки, и учитываем эффекты ускорения жидкости лишь за счёт давления. Определяем для крупной частицы промежуточные значения скоростей и энергий.

II. Лагранжев этап: вычисление потоков массы через границы эйлеровых ячеек.

III. Заключительный этап: определение окончательных значений газодинамических параметров на следующем временном шаге на основе законов сохранения массы импульса и энергии для каждой ячейки и всей системы в целом на фиксированной расчётной сетке.

Моделирующая среда заменяется системой из N частиц (жидких — «крупных» — частиц для сплошной среды и «молекул для дискретной), которые распределены в начальный момент времени по ячейкам эйлеровой сетки в координатном пространстве в соответствии с начальными данными.

Эволюция такой системы за время Δt осуществляется путём следующего расщепления: вначале изучается изменение внутреннего состояния подсистем, находящихся в ячейках, в предположении их замороженности или неподвижности (эйлеров этап для сплошной среды и столкновительная релаксация для дискретной), а затем рассматривается смещение всех частиц, пропорционально их скорости и Δt, без изменения внутреннего состояния подсистем с последующим пересчётом расчётной сетки в исходное положение (лагранжев и заключительный этапы для сплошной среды и свободное движение молекул для дискретной).

Рассмотрим движение идеального сжимаемого газа. В качестве исходных возьмём дифференциальные уравнения Эйлера в дивергентном виде (уравнения неразрывности, импульса, энергии):

$$\frac{\partial \rho_1}{\partial t} + \frac{\partial \rho_1 v_1}{\partial x} = 0, \qquad \frac{\partial \rho_2}{\partial t} + \frac{\partial \rho_2 v_2}{\partial x} = 0, \qquad \frac{\partial n}{\partial t} + \frac{\partial n v_2}{\partial x} = 0,$$

$$\rho_1 \frac{d_1 v_1}{dt} = -\alpha_1 \frac{\partial p}{\partial x} - nf, \qquad \rho_2 \frac{d_2 v_2}{dt} = -\alpha_2 \frac{\partial p}{\partial x} + nf,$$

$$\rho_1 \frac{d_1 e_1}{dt} = -\frac{p\alpha_1}{\rho_1^{\circ}} \frac{\partial \rho_1^{\circ}}{\partial t} + nf(v_1 - v_2) - nq,$$

$$\rho_2 \frac{d_2 e_2}{dt} = -\frac{p\alpha_2}{\rho_2^{\circ}} \frac{d\rho_1^{\circ}}{dt} + nq,$$

Где

$$f = \frac{\pi d^2 \rho_1^{\circ} C_d (v_1 - v_2)|v_1 - v_2|}{8} \qquad q = \pi d \lambda_1 Nu(T_1 - T_2)$$

Для замыкания системы уравнений используем

$$p = p_1(\rho_1^{\circ}, T_1) = p_2(\rho_2^{\circ}, T_2) \qquad e_1 = e_1(\rho_1^{\circ}, T_1) \qquad e_2 = e_2(\rho_2^{\circ}, T_2)$$

$$\rho_1 = \overset{\circ}{\rho_1}\alpha_1 \qquad \rho_2 = \overset{\circ}{\rho_2}\alpha_2 \qquad \alpha_1 + \alpha_2 = 1 \qquad E_i = e_i + \frac{{v_i}^2}{2}\ (i = 1,2)$$

При исследовании сложных структур течений будем использовать однородные разностные схемы, позволяющие проводить устойчивый сквозной счёт без предварительного выделения особенностей.

Условия однородности схемы и устойчивости вычислительного процесса требуют наличия некоторого диссипативного механизма в разностной схеме, что и приводит к сглаживанию фронта разрывов. В данной работе, члены, реализующие псевдовязкость, в явном виде вводятся в уравнения.

Характерной чертой прямой аппроксимации уравнений газовой динамики, записанных в виде законов сохранения, является свойство консервативности или дивергентности получаемых при этом разностных схем. Использование таких схем позволяет повысит точность вычислений, так как для них выполняются разностные законы сохранения как для отдельной ячейки, так и для всей системы в целом.

3.2. Предварительные преобразования

Для того, чтобы воспользоваться идеологией метода крупных частиц, необходимо привести описанную выше систему уравнений к дивергентному виду [2]:

$$v_1\left[\frac{\partial \rho_1 v_1}{\partial t} + \frac{\partial \rho_1 {v_1}^2}{\partial x}\right] = -v_1\alpha_1\frac{\partial p}{\partial x} - nfv_1,$$

$$v_2\left[\frac{\partial \rho_2 v_2}{\partial t} + \frac{\partial \rho_2 {v_2}^2}{\partial x}\right] = -v_2\alpha_2\frac{\partial p}{\partial x} + nfv_2,$$

$$\frac{\partial \rho_1 e_1}{\partial t} + \frac{\partial \rho_1 e_1 v_1}{\partial x} = \frac{p\alpha_1}{\rho_1^0}\frac{d_1\rho_1^0}{dt} + nf(v_1 - v_2) - nq,$$

$$\frac{\partial \rho_2 e_2}{\partial t} + \frac{\partial \rho_2 e_2 v_2}{\partial x} = \frac{p\alpha_2}{\rho_2^0}\frac{d_2\rho_2^0}{dt} + nq,$$

Рассмотрим все этапы расчётного цикла в отдельности.

3.3. Эйлеров этап.

На этом этапе расчёта изменяются лишь величины, относящиеся к ячейке в целом, а жидкость предполагается заторможенной. Поэтом конвективные члены вида $\frac{\partial f \rho v}{\partial x}$, где $f = (1, v, E)$, соответствующие эффектам перемещения, в системе уравнений отбрасываются.

Из уравнения неразрывность следует, что поле плотности будет заморожено, поэтому в оставшихся уравнениях можно вынести ρ из под знака дифференциала:

$$\frac{\partial \rho_1}{\partial t} = 0, \qquad \frac{\partial \rho_2}{\partial t} = 0, \qquad \frac{\partial n}{\partial t} = 0,$$

$$\rho_1 \frac{\partial v_1}{\partial t} = -\alpha_1 \frac{\partial p}{\partial x} - nf, \qquad \rho_2 \frac{\partial v_2}{\partial t} = -\alpha_2 \frac{\partial p}{\partial x} + nf,$$

$$\rho_1 \frac{\partial e_1}{\partial t} = -p\left(\frac{\partial \alpha_1 v_1}{\partial x} + \frac{\partial \alpha_2 v_2}{\partial x}\right) + nf(v_1 - v_2) - nq,$$

$$\rho_2 \frac{d_2 e_2}{dt} = nq,$$

Представленные выше выражения можно, посредством преобразований, предложенных в [5], привести к виду:

$$\frac{\partial \rho_1 \frac{{v_1}^2}{2}}{\partial t} + \frac{\partial \rho_1 v_1 \frac{{v_1}^2}{2}}{\partial x} = -v_1 \alpha_1 \frac{\partial p}{\partial x} - nf v_1,$$

$$\frac{\partial \rho_2 \frac{{v_2}^2}{2}}{\partial t} + \frac{\partial \rho_2 v_2 \frac{{v_2}^2}{2}}{\partial x} = -v_2 \alpha_2 \frac{\partial p}{\partial x} + nf v_2,$$

$$\frac{\partial \rho_1 e_1}{\partial t} + \frac{\partial \rho_1 e_1 v_1}{\partial x} = -p\left(\frac{\partial \alpha_1 v_1}{\partial x} + \frac{\partial \alpha_2 v_2}{\partial x}\right) + nf(v_1 - v_2) - nq,$$

$$\frac{\partial \rho_2 e_2}{\partial t} + \frac{\partial \rho_2 e_2 v_2}{\partial x} = nq,$$

Аппроксимируя уравнения в момент времени t^n, получим следующие разностные уравнения для ячейки (крупной частицы) i:

$$\rho_j{}_i^n \left(\frac{\partial v_j}{\partial t}\right)_i^n \Delta x = -\left[p^n_{i+\frac{1}{2}} - p^n_{i-\frac{1}{2}}\right]$$

$$\rho_j{}_i^n \left(\frac{\partial E_j}{\partial t}\right)_i^n \Delta x = -\left[p^n_{i+\frac{1}{2}} v_j{}^n_{i+\frac{1}{2}} - p^n_{i-\frac{1}{2}} v_j{}^n_{i-\frac{1}{2}}\right], (j = 1,2)$$

где производные по времени определяются так:

$$\left(\frac{\partial f}{\partial t}\right)_i^n = \frac{\tilde{f}_i^n - f_i^n}{\Delta t}, f = (v, E).$$

В результате получим для системы уравнений явные конечно-разностные уравнения первого порядка точности по времени и второго порядка по пространству

$$\tilde{v}_{1,i}^n = v_{1,i}^n - \frac{p_{i+\frac{1}{2}}^n - p_{i-\frac{1}{2}}^n}{\Delta x}\frac{\Delta t}{\rho_{1\,i}^{0\,n}} - \frac{n_i^n}{\rho_{1\,i}^{\,n}} f_i^m \Delta t$$

$$\tilde{v}_{2,i}^n = v_{2,i}^n - \frac{p_{i+\frac{1}{2}}^n - p_{i-\frac{1}{2}}^n}{\Delta x}\frac{\Delta t}{\rho_{2\,i}^{0\,n}} + \frac{n_i^n}{\rho_{2\,i}^{\,n}} f_i^m \Delta t$$

Используем ещё одну модификацию, связанную с частично неявной аппроксимацией силы межфазного взаимодействия. Введём обозначения:

$$\Delta p_{1,i}^{n+1} = \frac{1}{\rho_{1\,i}^{0\,n}}\left(p_{i+\frac{1}{2}}^n - p_{i-\frac{1}{2}}^n\right)\frac{\Delta t}{\Delta x}; \qquad \Delta p_{2,i}^{n+1} = \frac{1}{\rho_{2\,i}^{0\,n}}\left(p_{i+\frac{1}{2}}^n - p_{i-\frac{1}{2}}^n\right)\frac{\Delta t}{\Delta x};$$

$$\Delta f_{1,i}^n = \frac{n_i^n}{\rho_{1\,i}^{\,n}}\left(\frac{\pi d^2 \rho_1^{\circ} C_d |v_1 - v_2|}{8}\right)_i^n \Delta t; \quad \Delta f_{2,i}^n = \frac{n_i^n}{\rho_{2\,i}^{\,n}}\left(\frac{\pi d^2 \rho_1^{\circ} C_d |v_1 - v_2|}{8}\right)_i^n \Delta t;$$

Перепишем учётом указанных обозначений:

$$\tilde{v}_{1,i}^n = v_{1,i}^n - \Delta p_{1,i}^{n+1} - \Delta f_{1,i}^n\left(\tilde{v}_{1,i}^n - \tilde{v}_{2,i}^n\right)$$

$$\tilde{v}_{2,i}^n = v_{2,i}^n - \Delta p_{2,i}^{n+1} + \Delta f_{2,i}^n\left(\tilde{v}_{1,i}^n - \tilde{v}_{2,i}^n\right)$$

$$\tilde{v}_{1,i}^n - \tilde{v}_{2,i}^n = \frac{\left(v_{1,i}^n - v_{2,i}^n\right) - \left(\Delta p_{1,i}^{n+1} - \Delta p_{2,i}^{n+1}\right)}{1 + \left(\Delta f_{1,i}^n - \Delta f_{2,i}^n\right)}$$

Теперь можно определить промежуточные значения внутренней энергии конденсированной фазы:

$$\tilde{e}_{2,i}^n = e_{2,i}^n + n_i^n q_i^n \frac{\Delta t}{\rho_{2,i}^n};$$

Значение полной энергии:

$$\rho_{1_i}^{\,n}\tilde{E}_{1,i}^{n} + \rho_{2_i}^{\,n}\tilde{E}_{2,i}^{n}$$

$$= \rho_{1_i}^{\,n}E_{1,i}^{n} + \rho_{2_i}^{\,n}E_{2,i}^{n} - \left(\alpha_{1}{}_{i+\frac{1}{2}}^{\,n}\tilde{v}_{1,i+\frac{1}{2}}^{n}p_{i+\frac{1}{2}}^{n} - \alpha_{1}{}_{i-\frac{1}{2}}^{\,n}\tilde{v}_{1,i-\frac{1}{2}}^{n}p_{i-\frac{1}{2}}^{n}\right)\frac{\Delta t}{\Delta x}$$

$$-\left(\alpha_{2}{}_{i+\frac{1}{2}}^{\,n}\tilde{v}_{2,i+\frac{1}{2}}^{n}p_{i+\frac{1}{2}}^{n} - \alpha_{2}{}_{i-\frac{1}{2}}^{\,n}\tilde{v}_{2,i-\frac{1}{2}}^{n}p_{i-\frac{1}{2}}^{n}\right)\frac{\Delta t}{\Delta x}$$

Величины с дробными индексами, относящиеся к границам ячеек, находятся так:

$$v_{i+\frac{1}{2}}^{n} = \frac{v_i^n + v_{i+1}^n}{2}$$

В приведённых схемах $\tilde{v},\ \tilde{E}$ — промежуточные значения параметров потока на слое $t^n + \Delta t$, полученные в предположении заторможенности поля плотности без учёта эффектов перемещения среды.

Для улучшения устойчивости счёта первого этапа в уравнениях вместо p используется p+q. Здесь q — искусственное вязкостное давление (псевдовязкость):

$$q_{i+\frac{1}{2}} = \begin{cases} 0, \text{если } v_{i+1} - v_i \geq 0 \\ \rho_{1}{}_{i+\frac{1}{2}}^{\,n}\left(ac_{i+\frac{1}{2}}^{n} + b(v_{i+1} - v_i)\right), \text{если } v_{i+1} - v_i < 0 \end{cases}$$

3.4. Лагранжев этап

На данном этапе вычисляются эффекты переноса, учитывающие обмен между ячейками при их перестройке на прежнюю эйлерову сетку. Здесь находятся потоки массы $\Delta M_1^n, \Delta M_2^n$ и поток частиц через границы эйлеровых ячеек за время Δt. При этом полагаем, что вся масса переносится только за счёт нормальной к границе составляющей скорости. Например, уравнение неразрывности системы в разностной форме можно записать так:

$$\rho_{1_i}^{\,n+1}\Delta x = \rho_{1_i}^{\,n}\Delta x - \Delta M_{1}{}_{i+\frac{1}{2}}^{\,n} + \Delta M_{1}{}_{i-\frac{1}{2}}^{\,n}$$

$$\rho_{2_i}^{\,n+1}\Delta x = \rho_{2_i}^{\,n}\Delta x - \Delta M_{2}{}_{i+\frac{1}{2}}^{\,n} + \Delta M_{2}{}_{i-\frac{1}{2}}^{\,n}$$

Поток массы $\Delta M_{j\,i+\frac{1}{2}}^{n}$ $(j = 1,2)$ можно определить по формуле первого порядка точности:

$$\Delta M_{j\,i+\frac{1}{2}}^{n} = \begin{cases} \rho_{j\,i}^{n} \dfrac{\tilde{v}_i^n + \tilde{v}_{i+1}^n}{2} \Delta t, \tilde{v}_i^n + \tilde{v}_{i+1}^n > 0 \\ \rho_{j\,i+1}^{n} \dfrac{\tilde{v}_i^n + \tilde{v}_{i+1}^n}{2} \Delta t, \tilde{v}_i^n + \tilde{v}_{i+1}^n < 0 \end{cases} \quad j = 1,2$$

3.5. Заключительный этап

Здесь происходит перераспределение массы, импульса и энергии по пространству и определяются окончательные поля эйлеровых параметров потока на фиксированной сетке в момент времени $t^{n+1} = t^n + \Delta t$.

Как уже отмечалось, уравнения этого этапа представляют собой законы сохранения массы M, импульса P и полной энергии E, записанные для данной ячейки в разностной форме:

$$M^{n+1} = M^n + \sum \Delta M_{\text{гр}}^n$$
$$P^{n+1} = P^n + \sum \Delta P_{\text{гр}}^n$$
$$E^{n+1} = E^n + \sum \Delta E_{\text{гр}}^n$$

Здесь $\Delta M_{\text{гр}}^n$ — масса газа, которая пересекла за время Δt одну из границ рассматриваемой ячейки; суммирование производится по всем сторонам ячейки. Аналогичным образом понимаются $\sum \Delta P_{\text{гр}}^n$ и $\sum \Delta E_{\text{гр}}^n$.

Уравнения утверждают, в частности, что внутри поля течения не источников и стоков для M, P и E, а их изменение за время Δt осуществляется только за счёт взаимодействия на внешней границу области течения. При этом полагается, что потоки массы через границы ячеек, определяемые на втором этапе, несут промежуточные значения скорости и удельной энергии, определяемые на первом этапе. Величины $\Delta M_{j\,i+\frac{1}{2}}^{n}$ $(j = 1,2)$ играют здесь роль весовых функций.

Исходя из этого, окончательные значения параметров потока $\rho, X = (v_1, v_2, E_1, E_2)$ на новом временном слое $t^{n+1} = t^n + \Delta t$ вычисляются по формулам

$$\rho_{j_i}^{n+1} = \rho_{j_i}^{n} - \frac{\Delta M_{j}{}^{n}_{i+\frac{1}{2}} - \Delta M_{j}{}^{n}_{i-\frac{1}{2}}}{\Delta x},$$

$$X_{j_i}^{n+1} = \frac{\rho_{j_i}^{n}}{\rho_{j_i}^{n+1}} \tilde{X}_{j_i}^{n} - \frac{\tilde{X}_{j}{}^{n}_{i+1} \Delta M_{j}{}^{n}_{i+\frac{1}{2}} - \tilde{X}_{j}{}^{n}_{i-1} \Delta M_{j}{}^{n}_{i-\frac{1}{2}}}{\Delta x}, \quad (j = 1,2)$$

Формула справедлива, когда поток втекает в ячейку i через все стороны.

Таким образом, каждый из всех трёх этапов имеет ряд трактовок. Комбинируя различные представления на этапах, получаем серию разностных схем модифицированного метода крупных частиц, что и позволяет осуществить широкий класс численных экспериментов.

4. РЕЗУЛЬТАТЫ

Расчёты проводились для смеси воздуха с частицами кварцевого песка со следующими параметрами:

$$T_0 = 293°K, \qquad p_0 = 0.1\ \text{МПа}, \qquad c_p = 1005\,\frac{\text{м}^2}{\text{с}^2 \cdot °\text{К}},$$

$$\gamma = 1.4, \qquad a_{10} = 341\,\frac{\text{м}}{\text{с}}, \qquad \rho_{10}^0 = 1.21\,\frac{\text{кг}}{\text{м}^3},$$

$$\lambda_1 = 0.026\,\frac{\text{кг} \cdot \text{м}}{\text{с}^3 \cdot °\text{К}}, \quad \mu_1 = 1.85 \cdot 10^{-5}\,\frac{\text{кг}}{\text{м} \cdot \text{с}}, \qquad \rho_2^0 = 2500\,\frac{\text{кг}}{\text{м}^3},$$

$$c_2 = 710\,\frac{\text{м}^2}{\text{с}^2 \cdot °\text{К}}$$

Расстояние от фронта ударной волны до неподвижной стенки задавалось равным 3.05 м, длина волны 0.45 м, число Маха ударной волны 4.2. Начальное распределение давления представлено на (Рис. 1).

Профили давления в разные моменты времени при протекании процесса в чистом газе представлен на рисунке 1 и рисунке 2. Профили давления ударной волны, взаимодействующей с экранирующим слоем газовзвеси, в разные моменты времени представлен на рисунках 3 и 4. Расчёт выполнен для экранирующего слоя газовзвеси с длиной $l_s = 3$м, объёмных долей частиц $0.001 \leq \alpha_2 \leq 0.05$ и диаметром частиц 60 мкм $\leq d \leq 1$ мм. Можно увидеть, что наличие экранирующего слоя приводит к заметной трансформации падающей на стенку нестационарной ударной волны: волна становится более слабой и более длинной, а скорость несущей фазы значительно снижается. При объёмной доле частиц $\alpha_2 > 0.01$ возникает явная отражённая от частиц волна.

В процесс взаимодействия ударной волны с неподвижно стенкой можно выделить три этапа: мгновенное повышение давления на стенке, последующее плавное повышение давления и постепенное падение давления до начального. Первый этап связан с отражением от стенки переднего ударного скачка в газовой фазе, при этом давление в ней резко повышается, возникает движущаяся в обратном направлении отражённая

ударная волна. На втором этапе давление повышается плавно в силу взаимодействия падающей и отражённой ударных волн. Отражённая ударная волна движется навстречу набегающему потоку газовзвеси, давление которого плавно возрастает, а скорость уменьшается. Наибольшее давление образуется вблизи стенки, где тормозится набегающий поток. Происходит дальнейшее взаимодействие отражённой волны уплотнения с волной разряжения, что подводит к началу третьего этапа, этапа разгрузки преграды. Давление на стенке начинает уменьшаться, а поток меняет своё направление на противоположное. Первой своё направление меняет несущая, газовая фаза, постепенно увлекая за собой дисперсную, если масса дисперсной фазы достаточно мала, иначе вызывая снижение скорости. Тяжёлые частицы, движущиеся медленнее, чем газ, или движущиеся в противоположную сторону, препятствуют оттоку газа от стенки, увеличивая тем самым время действия избыточного давления на неё.

На рисунках 5 и 6 проиллюстрирована зависимость профилей давления ударной волны прошедшей расстояние 1.05 м, из которого 1 м заполнен экранирующим слоем с объёмными долями частиц $\alpha_2 = 0.001$ и $\alpha_2 = 0.01$ соответственно, от диаметра частиц. Из приведённых графиков видно, как трансформируется при уменьшении диаметра частиц ударная волна, снижается максимальное давление и становится всё более явной отражённая ударная волна.

Зависимости времени прохождения ударной волной указанного расстояния от объёмной доли при фиксированном диаметре частиц приведена на Рис. 7. Из Рис. 7 видно, что при увеличении объёмной доли частиц и уменьшении их диаметра эффективность снижения скорости набегающей ударной волны экранирующим слоем газовзвеси повышается.

Проведён анализ взаимодействия набегающей ударной волны с неподвижной стенкой путём вычисления импульсов избыточного давления

и скоростного напора частиц, а также максимальных значений избыточного давления и скоростного напора частиц.

Величину импульсного воздействия потока газа с частицами можно характеризовать суммой

$$I(t) = I_1(t) + I_2(t)$$

$$I_1(t) = \int_{t_*}^{t} (p_w - p_0) d\tau \, , I_2(t) = \int_{t_*}^{t} \frac{\rho_{2,w} {u_{2,w}}^2}{2} d\tau.$$

Здесь $I_1(t)$ – импульс избыточного давления газа, а $I_2(t)$ – импульс скоростного напора частиц.

На Рис. 8 и Рис. 10 приведены графики значений избыточного давления для объёмных долей частиц $\alpha_2 = 0.001$ и $\alpha_2 = 0.01$ соответственно. Из графиков видно, что уменьшение диаметра приводит как к уменьшению максимального давления, создаваемого набегающей ударной волной на стенке, так и к увеличению времени разгрузки преграды, что приводит к увеличению импульса избыточного давления.

На Рис. 9 и Рис. 11 приведены графики значений давления скоростного напора частиц для объёмных долей частиц $\alpha_2 = 0.001$ и $\alpha_2 = 0.01$ соответственно. Графики демонстрируют увеличение максимальных величин скоростного напора с увеличением объёмной доли и уменьшением диаметра частиц. Как видно, увеличение объёмной доли до 0.001 приводит к значительному увеличению величины скоростного напора при большом диаметре частиц, но уменьшение диаметра частиц приводит к снижению этой величины практически до нуля.

Как видно из графиков, характерное время взаимодействия ударной волны со стенкой при увеличении массовой доли частиц так же увеличивается.

Значение величин $I_1(t)$ и $I_2(t)$ приведены в Таблице 1.

Таблица 1

α_2		1 мм	0.6 мм	0.3 мм	0.1 мм	60 мкм
0.001	I_1	2057	2007	1996	2158	2272
	I_2	7.62	11.68	15.12	13.96	11.09
	I	2065	2019	2011	2172	2283
0.01	I_1	1194	1495	1155	1250	1298
	I_2	281	255	160	12	2
	I	1475	1750	1315	1262	1300

В таблице 2 приведены максимальное избыточное давление $P_{1,max}$, максимальная величина скоростного напора частиц $P_{2,max}$ и суммарное максимальное давление оказываемое на стенку. Вычисления проводились по следующим формулам:

$$P_{max} = \max_{\tau\in[t_*;t]}(p_w(\tau) - p_0 + \frac{\rho_{2,w}(\tau){u_{2,w}}^2(\tau)}{2})$$

$$P_{1,max} = \max_{\tau\in[t_*;t]}(p_w(\tau) - p_0)\,, P_{2,max} = \max_{\tau\in[t_*;t]}\left(\frac{\rho_{2,w}(\tau){u_{2,w}}^2(\tau)}{2}\right)$$

При протекании процесса в чистом газе эти величины принимают следующее значение:

$P_{max}(\alpha_2 = 0) = P_{1,max}(\alpha_2 = 0) = 1276500$ Па, $\quad P_{2,max}(\alpha_2 = 0) = 0$ Па

Таблица 2

α_2		1 мм	0.6 мм	0.3 мм	0.1 мм	60 мкм
0.001	$P_{1,max}$	925130	772770	537270	233930	203740
	$P_{2,max}$	794	1286	1932	2496	2727
	P_{max}	925130	772770	537270	233930	203740
0.01	$P_{1,max}$	164580	88926	36653	34157	34838
	$P_{2,max}$	18501	15520	3588	127	11
	P_{max}	164580	88926	36653	34157	34838

Анализируя значения, приведённых в таблицах 1 и 2, можно увидеть, что увеличение объёмной доли дисперсной фазы приводит к значительному снижению как импульса, передаваемого на стенку, так и максимального избыточного давления, возникающего в области торможения несущей фазы вблизи стенки. При этом уменьшение диаметра частиц не даёт однозначного улучшения: максимальное избыточное давление на стенке заметно уменьшается, но импульс передаваемый на стенку уменьшается лишь до определённого момента, начиная с которого дальнейшее уменьшение диаметра частиц приводит к усилению влияния накопившихся возле стенки частиц (рисунок 12), что, в свою очередь, приводит к увеличению времени разгрузки преграды и, как следствие, увеличению импульса.

ЗАКЛЮЧЕНИЕ

В ходе выполнения работы была рассмотрена математическая модель двухскоростного двухтемпературного континуума [2]. На основе метода крупных частиц [4] был разработан численный алгоритм, реализующая сквозное интегрирование уравнений данной модели, реализация которого приведена в приложении 2. С использованием разработанного численного алгоритма было произведено моделирование набегания нестационарной ударной волны на неподвижную стенку, экранированную слоем однородной газовзвеси. Результаты моделирования с различными параметрами экранирующего слоя представлены в данной работе. Анализ результатов работы позволяет сравнить эффективность экранирования слоем газовзвеси с различными параметрами. Сравнение результатов с работой [3] показало хорошее совпадение. Результаты могут быть полезны при проектировании систем пылевой защиты наземных сооружений и персонала от воздействия ударных волн и анализе возможностей газодинамических методов нанесения порошковых покрытий.

ПРИЛОЖЕНИЕ 1

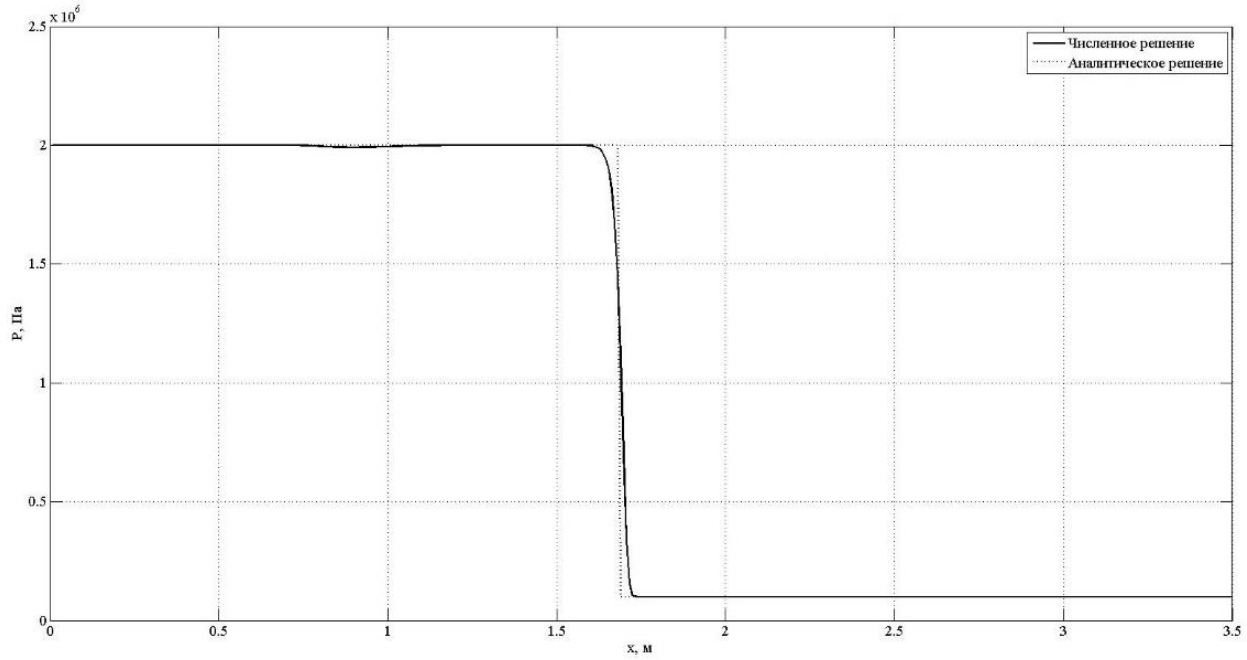

Рисунок 1. Сравнение численного и аналитического решений.

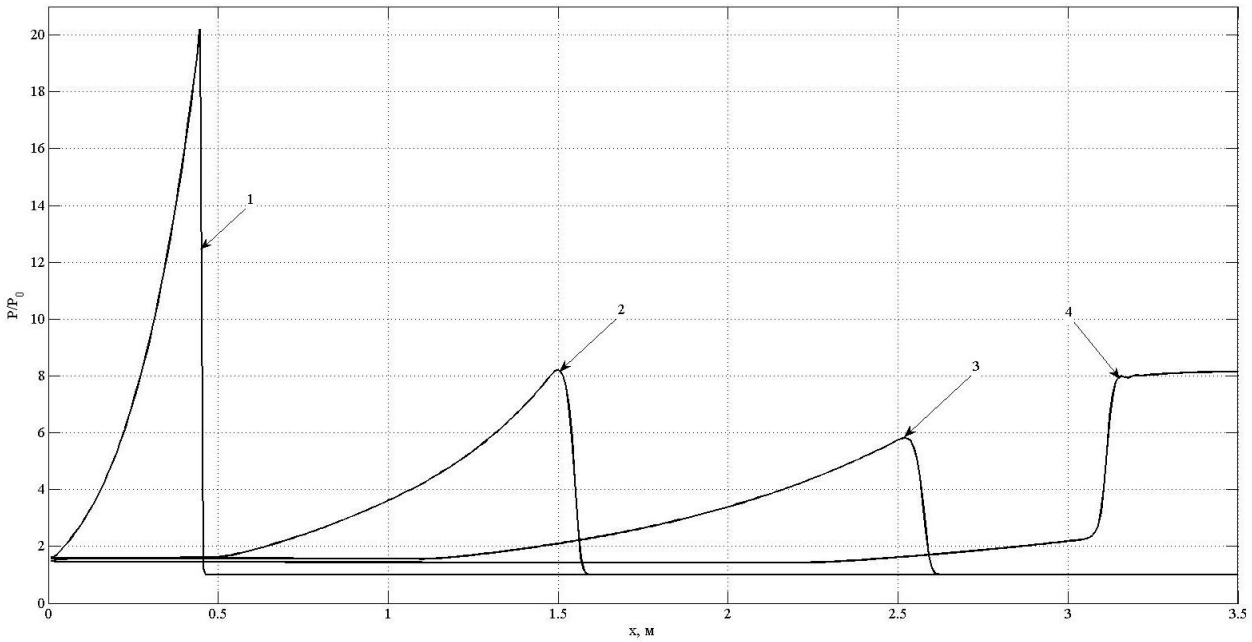

Рисунок 1 — Профили безразмерного давления в чистом газе в моменты времени 1 - $t = 0$ с, 2 - $t = 0.9$ мс, 3 - $t = 2.0$ мс, 4 - $t = 4.0$ мс.

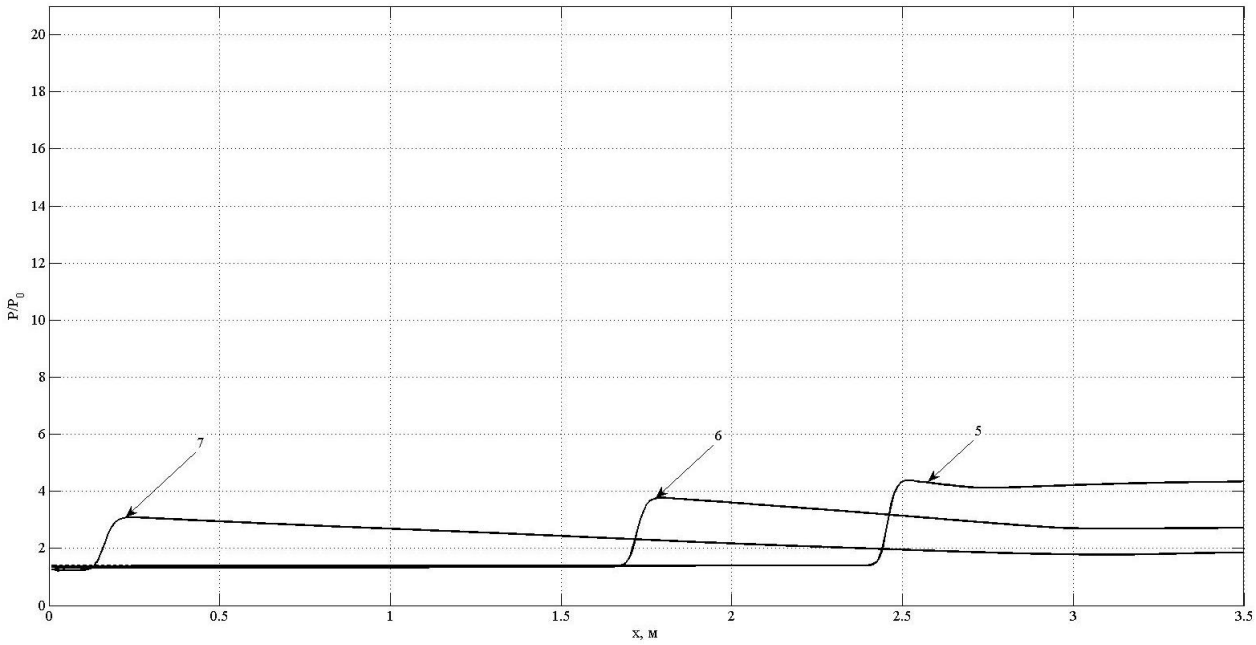

Рисунок 2 — Профили безразмерного давления в чистом газе в моменты времени 5 - $t = 5.0$ с, 6 - $t = 6.0$ мс, 7 - $t = 8.0$ мс.

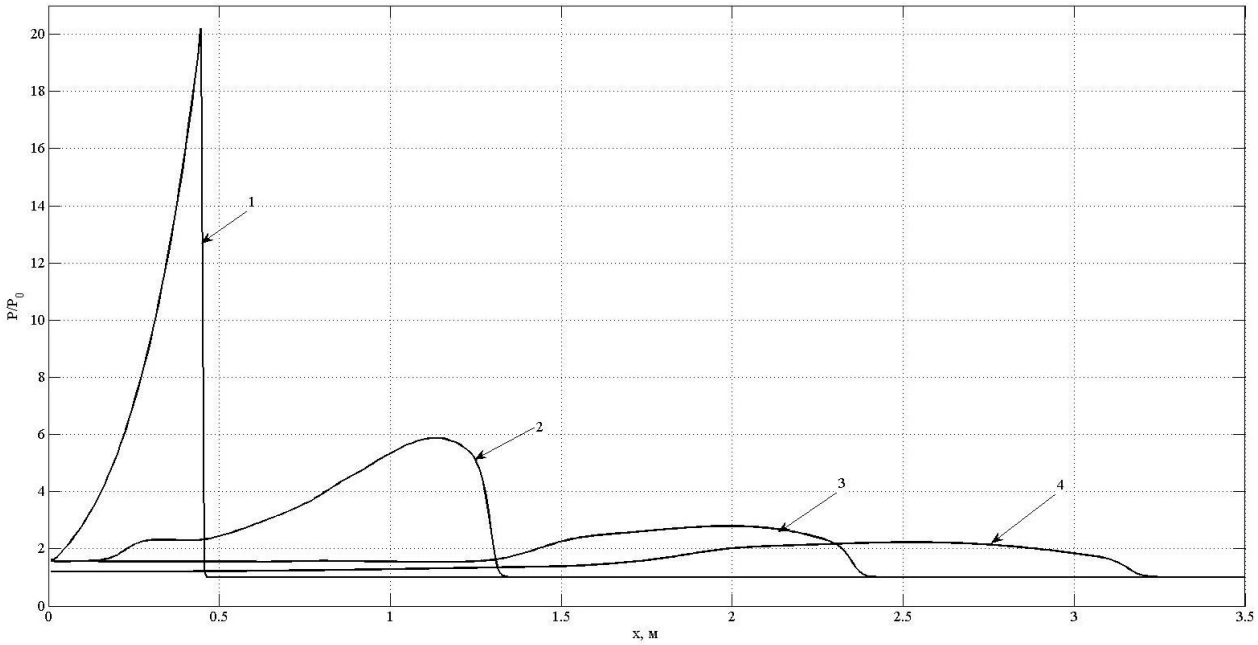

Рисунок 3 — Профили безразмерного давления в газовзвеси в моменты времени 1 - $t = 0$ с, 2 - $t = 0.9$ мс, 3 - $t = 2.8$ мс, 4 - $t = 4.6$ мс.

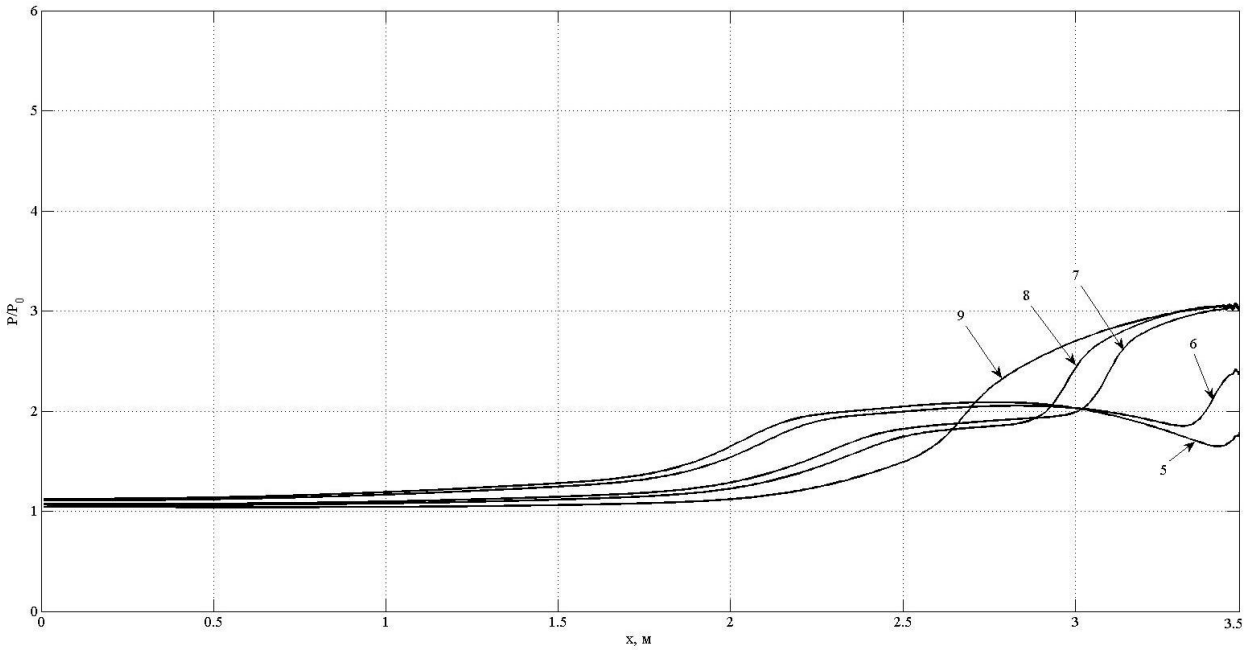

Рисунок 4 — Профили безразмерного давления в газовзвеси в моменты времени 5 - $t = 5.5$ мс, 6 - $t = 5.8$ мс, 7 - $t = 6.9$ мс, 8 - $t = 7.3$ мс, 8 - $t = 8.3$ мс.

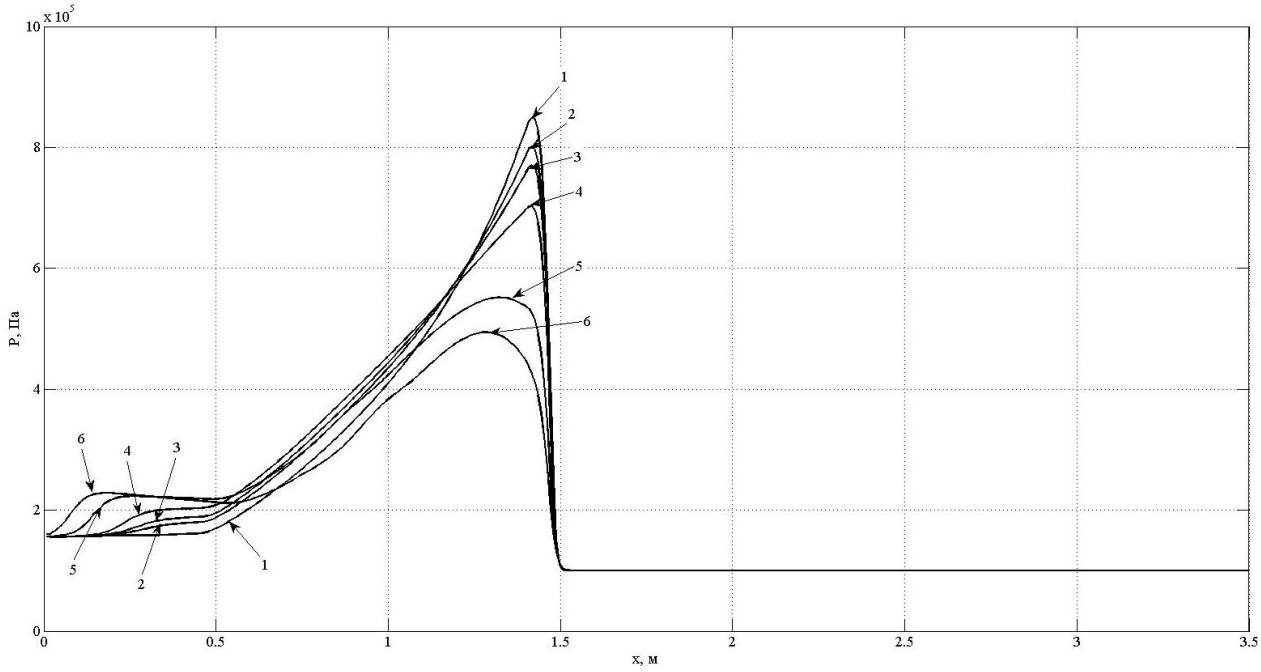

Рисунок 5 — Зависимость профиля давления ударной волны, прошедшей расстояние $l = 1.05$ м при $\alpha_2 = 0.001$ от диаметра частиц 1 – чистый газ, 2 – $d = 1$ мм, 3 – $d = 600$ мкм, 4 – $d = 300$ мкм, 5 – $d = 100$ мкм, 6 – $d = 60$ мкм.

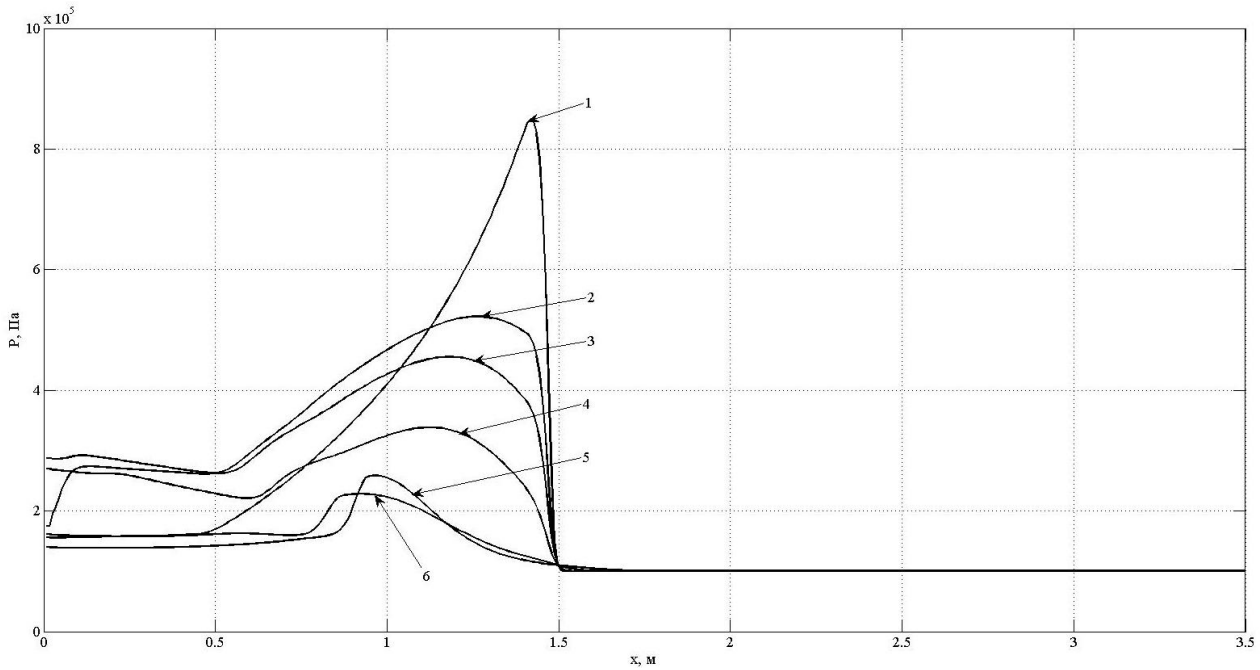

Рисунок 6 — Зависимость профиля давления ударной волны, прошедшей расстояние $l = 1.05$ м при $\alpha_2 = 0.01$ от диаметра частиц 1 – чистый газ, 2 – $d = 1$ мм, 3 – $d = 600$ мкм, 4 – $d = 300$ мкм, 5 – $d = 100$ мкм, 6 – $d = 60$ мкм.

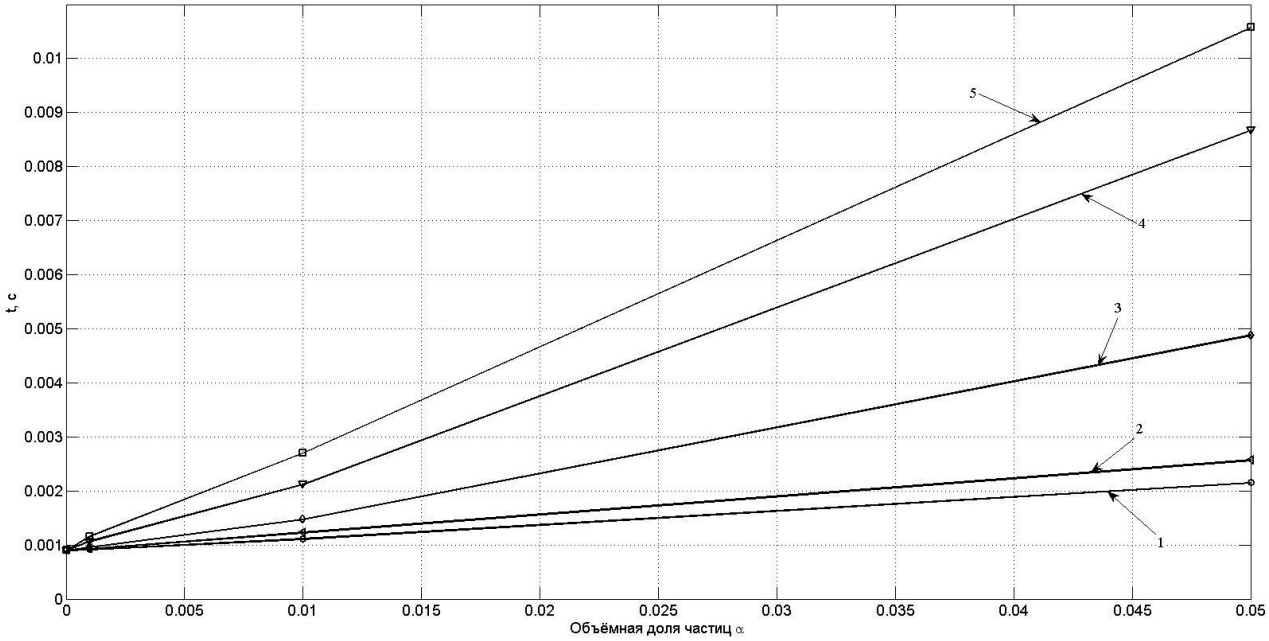

Рисунок 7 — Зависимость времени прохождения ударной волной расстояния $l = 1.5$ м от объёмной доли частиц для разных диаметров частиц: 1 – $d = 1$ мм, 2 – $d = 600$ мкм, 3 – $d = 300$ мкм, 4 – $d = 100$ мкм, 5 – $d = 60$ мкм.

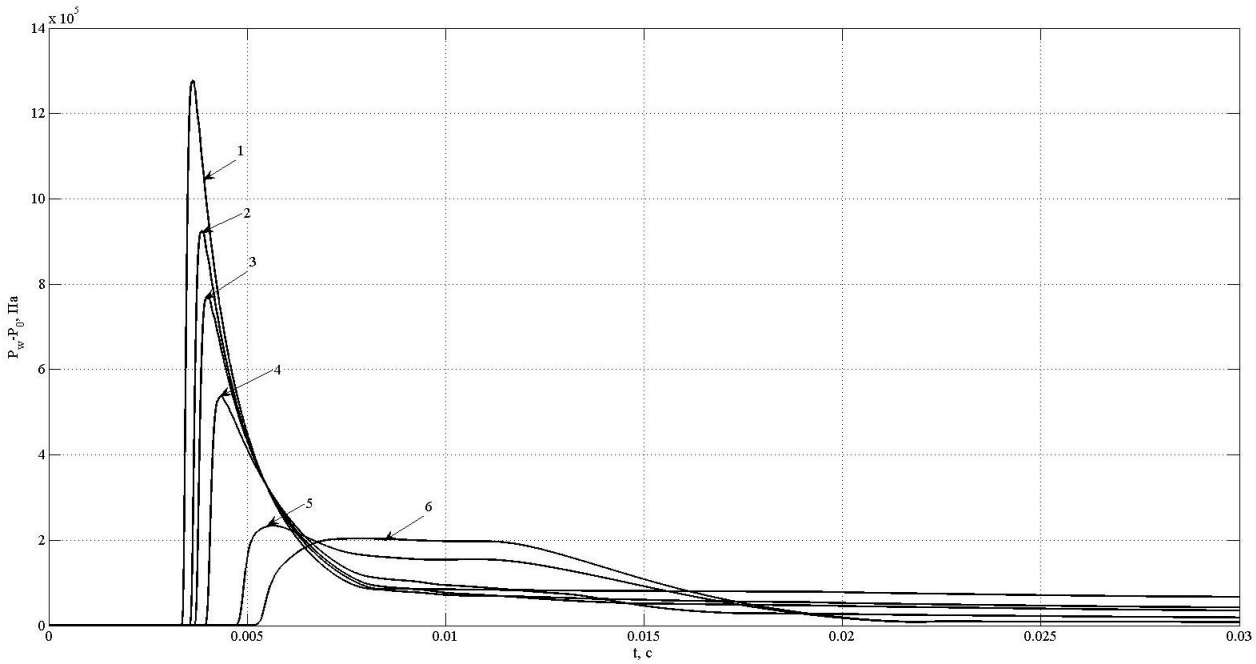

Рисунок 8 — Импульс избыточного давления, передаваемый на стенку с объёмным содержанием частиц $\alpha_2 = 0.001$ для разных частиц диаметром:1 – чистый газ, 2 – $d = 1$ мм, 3 – $d = 600$ мкм, 4 – $d = 300$ мкм, 5 – $d = 100$ мкм, 6 – $d = 60$ мкм.

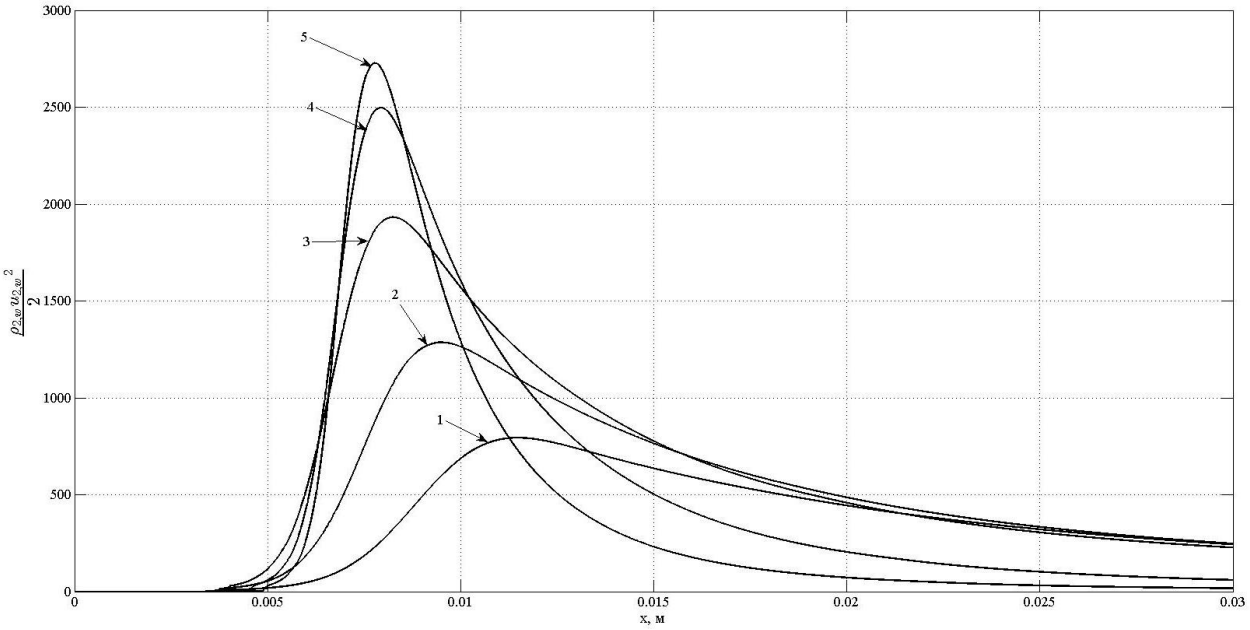

Рисунок 9 — Импульс скоростного напора, передаваемый на стенку с объёмным содержанием частиц $\alpha_2 = 0.001$ для разных частиц диаметром: 1 – $d = 1$ мм, 2 – $d = 600$ мкм, 3 – $d = 300$ мкм, 4 – $d = 100$ мкм, 5 – $d = 60$ мкм.

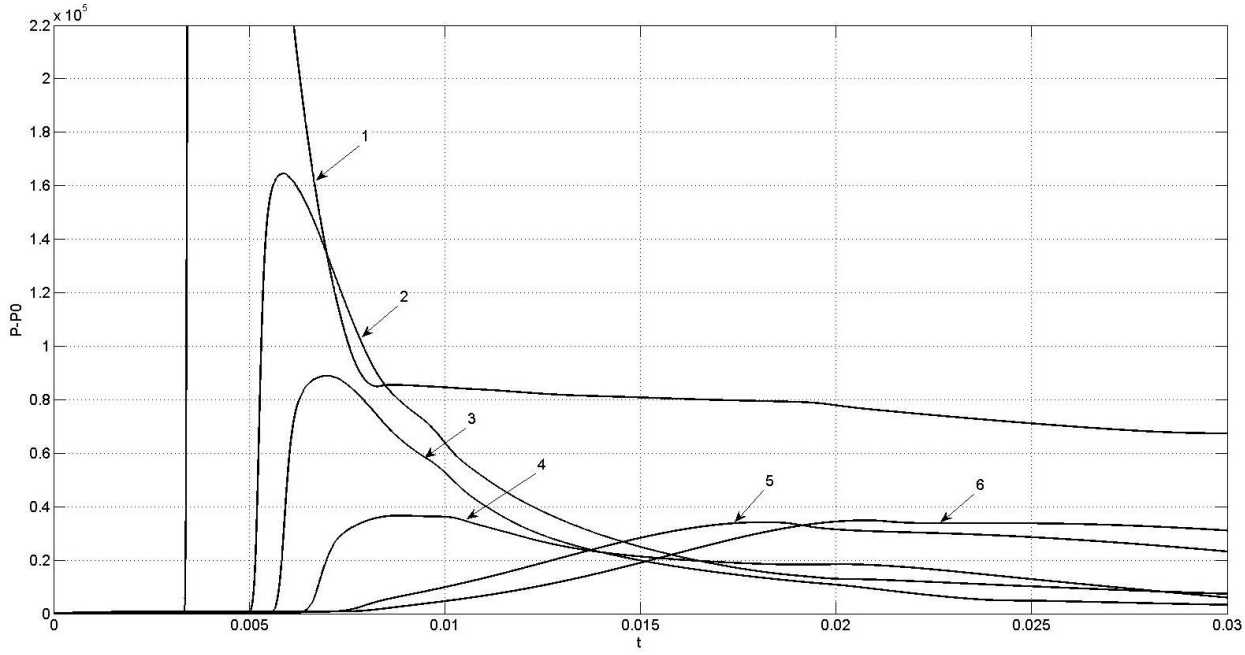

Рисунок 10 — Импульс избыточного давления, передаваемый на стенку с объёмным содержанием частиц $\alpha_2 = 0.01$ для разных частиц диаметром:1 – чистый газ, 2 – $d = 1$ мм, 3 – $d = 600$ мкм, 4 – $d = 300$ мкм, 5 – $d = 100$ мкм, 6 – $d = 60$ мкм.

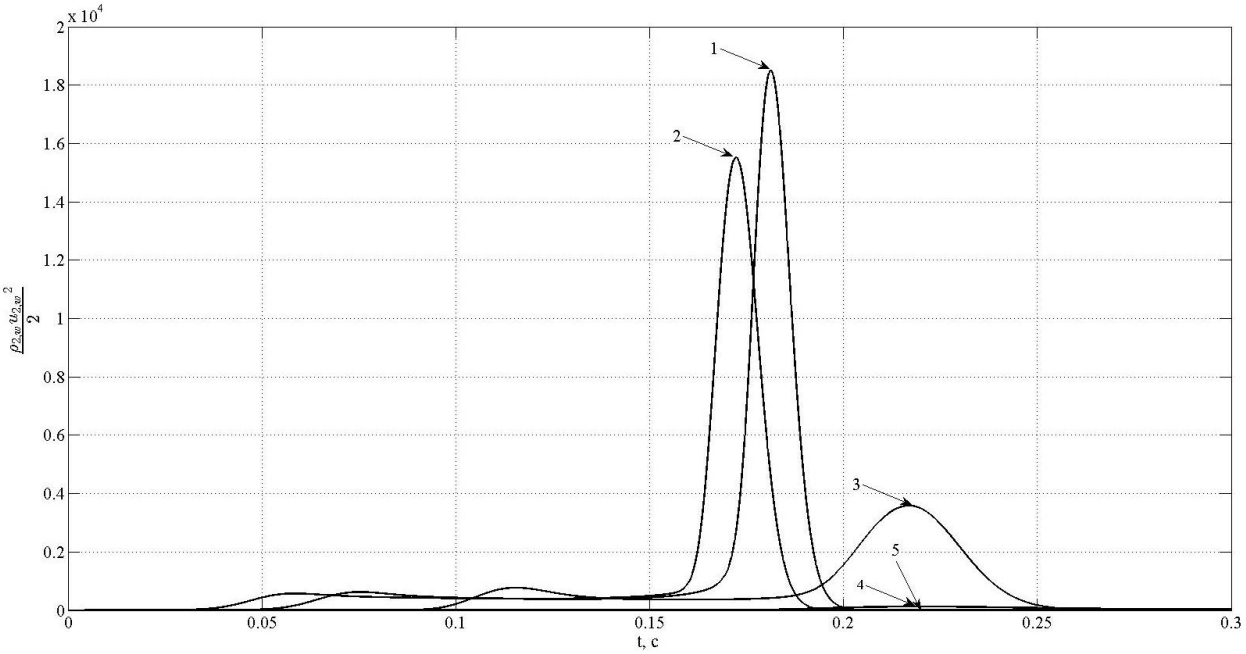

Рисунок 9 — Импульс скоростного напора, передаваемый на стенку с объёмным содержанием частиц $\alpha_2 = 0.01$ для разных частиц диаметром: 1 – $d = 1$ мм, 2 – $d = 600$ мкм, 3 – $d = 300$ мкм, 4 – $d = 100$ мкм, 5 – $d = 60$ мкм.

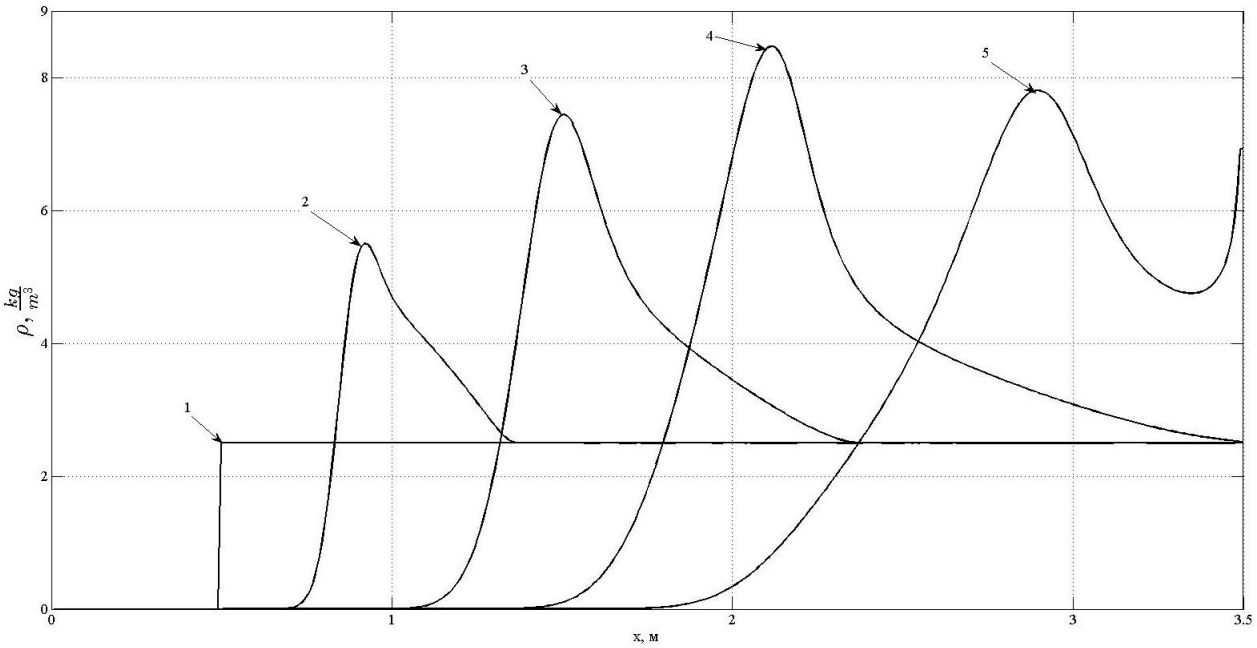

Рисунок 12 — Плотность конденсированной фазы в моменты времени 1 - $t = 0$ с, 2 - $t = 0.9$ мс, 3 - $t = 2.8$ мс, 4 - $t = 5.5$ мс, 5 - $t = 14.8$ мс.

ПРИЛОЖЕНИЕ 2

Исходный код:

```
#include <iostream>
#include <fstream>
#include <sstream>
#include <thread>
#include <math.h>
using namespace std;

void output(int N, double tstop, double alpha, double d, double dx, double *fn, string s);
double flowDirection(int index, double* VIntermediate, double* value);
double flowDirection(int index, double* VIntermediate, double* value, bool isFirstPhase);
void MassTranslation(double& RightBorderdM, double& LeftBorderdM, double V1, double Ro1, double leftCellVelocity1, double rightCellVelocity1, double leftCellDensity1, double rightCellDensity1, double dt, double dx, bool leftBorderAble, bool rightBorderAble);

int main()
{
	ofstream DebugLog;
	DebugLog.open("DebugLog.txt");
	int i = 0;
	const int N = 400;
	const double gamma = 1.4;
	const double PI = 3.1415926535;
	const int debugOutputFrequency = 10000;
	const double a = 0.9, b = 0.5;
	const double lambda1 = 0.026;
	const double d = 0.001;
	const double cp = 1005;
	const double cv = cp / 1.4;
	const double c2 = 710;
```

```
const double R = 287;
const double T0 = 293;
const double T2 = 293;
const double Mach = 4.2;
const double p0 = 100000;
const double p1 = (1.0 + (2.0 * gamma*(Mach*Mach - 1.0)) / (gamma + 1.0))*p0;
DebugLog << "p0 = " << p0 << " p1 = " << p1 << endl;
const double ro0 = 1.21;
const double ro2 = 2500;
const double alpha1 = 0.99;
const double ro1 = ((gamma + 1.0)*Mach*Mach / (2 + (gamma - 1.0)*Mach*Mach))*ro0;
DebugLog << "Ro0 = " << ro0 << " Ro1 = " << ro1 << endl;
const double etta0 = 1 / ro0;
const double etta1 = 1 / ro1;
const double c0 = sqrt(gamma*p0 *etta0);
const double c1 = sqrt(gamma*p1 *etta1);
DebugLog << "c0 = " << c0 << " c1 = " << c1 << endl;
const double u0 = sqrt(etta0*etta0*(p1 - p0) / (etta0 - etta1));
double v0 = 0;
double D = v0 - u0;
const double v1 = 2.0 / (gamma + 1.0)*(Mach - 1.0 / Mach)* c0;
DebugLog << "v0 = " << v0 << " v1 = " << v1 << endl;
const double e0 = p0 / (ro0*(gamma - 1)) + v0*v0 / 2.0;
const double e1 = p1 / (ro1*(gamma - 1)) + v1*v1 / 2.0;
DebugLog << "e0 = " << e0 << " e1 = " << e1 << endl;
const double Co = 0.01;
const double ls = 0.45;//длина волны
const double lw = 3.05;//рассто¤ние от волны до преграды
const double L = 3.5;
const double dx = L / N;
const double dt = Co*dx / (sqrt(c1*c1 + v1*v1));
double xi = 0;
const int OutputFrequency = int(0.0002 / dt);
```

```
cout << "OutputFrequency" << OutputFrequency << endl;
system("PAUSE");
double k = 0;
double mu = 1.85e-5;
double Pr = cp*mu / lambda1;
DebugLog << "dx = " << dx << " dt = " << dt << endl;
DebugLog << "k = " << k << " mu = " << mu << endl;
double tstop = 0.50;
double t = 0;
double alpha105;
double alpha205;
double maxPImpulse = 0;
double maxEPressure = 0;
double* temp = new double[N];
double* X = new double[N];
double* V = new double[N];
double* VIntermediate = new double[N];
double* V1 = new double[N];
double* U = new double[N];
double* UIntermediate = new double[N];
double* U1 = new double[N];
double* P = new double[N];
double* PIntermediate = new double[N];
double* P1 = new double[N];
double* Ro0_1 = new double[N];
double* Ro1_1 = new double[N];
double* Ro0_2 = new double[N];
double* Ro1_2 = new double[N];
double* RightBorderDensityFlow1 = new double[N];
double* RightBorderDensityFlow2 = new double[N];
double* RightBorderVelocityFlow1 = new double[N];
double* RightBorderVelocityFlow2 = new double[N];
double* RightBorderEnergyFlow1 = new double[N];
double* RightBorderEnergyFlow2 = new double[N];
double* RightBorderFullEnergyFlow2 = new double[N];
```

```
double* LeftBorderMassFlow1 = new double[N];
double* RightBorderMassFlow1 = new double[N];
double* LeftBorderMassFlow2 = new double[N];
double* RightBorderMassFlow2 = new double[N];
double* LeftBorderParticleFlow = new double[N];
double* RightBorderParticleFlow = new double[N];
double* E0_1 = new double[N];
double* EIntermediate_1 = new double[N];
double* E1_1 = new double[N];
double* E0_2 = new double[N];
double* EIntermediate_2 = new double[N];
double* E1_2 = new double[N];
double* e0_2 = new double[N];
double* e1_2 = new double[N];
double* S = new double[N];
double *viscousP = new double[N];
double *q1 = new double[N];
double *q2 = new double[N];
double *f = new double[N];
double *q = new double[N];
double *Re = new double[N];
double *Cd = new double[N];
double *Nu = new double[N];
double *alpha = new double[N];
double *n0 = new double[N];
double *n1 = new double[N];
double *dP1 = new double[N];
double *dP2 = new double[N];
double *df1 = new double[N];
double *df2 = new double[N];
double *dV = new double[N];

double* ExcessPressure2 = new double[int(tstop / (dt*100))];
double* ParticleImpulse3 = new double[int(tstop / (dt*100))];
string s = "";
```

```
//Initial fill
/*---------------------------------------------------------------------------*/
for (i = 0; i < N; i++)
{
	//Stat
	//if (i*dx < ls)
	//{
	//	alpha[i] = 1;
	//	U[i] = 0;
	//	V[i] = v1;
	//	xi = pow(1 - (gamma - 1)*(v1 - V[i]) / (2 * c1), 2 / (gamma - 1));
	//	P[i] = p1;
	//	Ro0_1[i] = ro1;
	//	Ro0_2[i] = 0;
	//	E0_2[i] = 0;
	//	E0_1[i] = P[i] / (Ro0_1[i] * (gamma - 1)) + V[i] * V[i] / 2.0;//e1;// for stac
	//}
	//else
	//{
	//	alpha[i] = 1;
	//	U[i] = 0;
	//	V[i] = 0;
	//	xi = pow(1 - (gamma - 1)*(v1 - V[i]) / (2 * c1), 2 / (gamma - 1));
	//	P[i] = p0;
	//	Ro0_1[i] = ro0;
	//	Ro0_2[i] = 0;
	//	E0_2[i] = 0;
	//	E0_1[i] = P[i] / (Ro0_1[i] * (gamma - 1)) + V[i] * V[i] / 2.0;//e1;// for stac
	//}
```

```
//Non-stat
if (i*dx < ls)
{
	alpha[i] = 1;
	U[i] = 0;
	V[i] = v1*i*dx / ls;
	xi = pow(1 - (gamma - 1)*(v1 - V[i]) / (2 * c1), 2 / (gamma - 1));
	P[i] = p1*pow(xi, gamma);
	Ro0_1[i] = ro1*xi;
	Ro0_2[i] = 0;
	E0_2[i] = 0;
	E0_1[i] = P[i] / (Ro0_1[i] * (gamma - 1)) + V[i] * V[i] / 2.0;//e1;// for stac
}
else
{
	if (i*dx > ls + 0.05)
	{
		alpha[i] = alpha1;
		U[i] = 0;
		V[i] = v0;
		P[i] = p0;
		Ro0_1[i] = ro0*alpha[i];
		Ro0_2[i] = ro2*(1 - alpha[i]);
		V[i] = v0;
		E0_1[i] = P[i] / (Ro0_1[i] * (gamma - 1)) + V[i] * V[i] / 2.0;
		E0_2[i] = c2*T2 + U[i] * U[i] / 2;

	}
	else
	{
		alpha[i] = 1;
		U[i] = 0;
```

```
                                V[i] = v0;
                                P[i] = p0;
                                Ro0_1[i] = ro0;
                                Ro0_2[i] = 0;
                                E0_1[i] = P[i] / (Ro0_1[i] * (gamma - 1)) + V[i] * V[i] /
2.0;//e1;// for stac
                                E0_2[i] = 0;
                        }
                }
                n0[i] = 6 * (1 - alpha[i]) / (PI*d*d*d);
                VIntermediate[i] = EIntermediate_1[i] = 0;
                X[i] = i*dx;
        }
        /*---------------------------------------------------------------------------*/
        int loopCount = 0;
        int outputLoopCount = 0;
        double n05, q05, f05;
        while (t < tstop)
        {
                for (i = 0; i < N; i++)
                {
                        Re[i] = (Ro0_1[i] / alpha[i])*fabs(V[i] - U[i])*d / mu;
                        Nu[i] = 2 + 0.6*pow(Pr, 1 / 3.0)*sqrt(Re[i]);
                        //Стандартный
                        Cd[i] = 0.42*fabs(V[i] - U[i]) + (1 - alpha[i])*mu / (Ro0_1[i] *
d)*(24 + 4.4*mu*sqrt(fabs(V[i] - U[i])*Ro0_1[i] * d / (alpha[i] * mu)));
                        f[i] = PI*d*d*Cd[i] * (Ro0_1[i] / alpha[i]);
                        q[i] = PI*d*lambda1*Nu[i] * ((E0_1[i] - V[i] * V[i] / 2) / cv -
(E0_2[i] - U[i] * U[i] / 2) / c2);
                }

                //Main pass
                for (i = 1; i < N - 1; i++)
                {
```

```
                    q1[i] = V[i + 1] < V[i] ? ((Ro0_1[i + 1] + Ro0_1[i]) / 2.0)*(a*sqrt(1.4*(P[i + 1] + P[i]) / (Ro0_1[i + 1] + Ro0_1[i])) + b*(V[i] - V[i + 1]))*(V[i] - V[i + 1]) : 0;
                    q2[i] = V[i - 1] > V[i] ? -((Ro0_1[i - 1] + Ro0_1[i]) / 2.0)*(a*sqrt(1.4*(P[i - 1] + P[i]) / (Ro0_1[i - 1] + Ro0_1[i])) - b*(V[i] - V[i - 1]))*(V[i] - V[i - 1]) : 0;
                }

                q1[0] = q1[1];
                q1[N - 1] = q1[N - 2];
                q2[0] = q2[1];
                q2[N - 1] = q2[N - 2];

                for (i = 0; i < N - 1; i++)
                {
                    alpha105 = (alpha[i] + alpha[i + 1]) / 2;
                    alpha205 = (2 - (alpha[i] + alpha[i + 1])) / 2;
                    n05 = (n0[i] + n0[i + 1]) / 2;
                    q05 = (q[i] + q[i + 1]) / 2;
                    f05 = (f[i] + f[i + 1]) / 2;
                    PIntermediate[i] = (P[i] + P[i + 1]) / 2.0;
                }
                PIntermediate[N - 1] = PIntermediate[N - 2];

                for (i = 1; i < N - 1; i++)
                {
                    dP1[i] = (PIntermediate[i] - PIntermediate[i - 1] + q1[i] - q2[i])*dt / (dx*(Ro0_1[i] / alpha[i]));
                    df1[i] = n0[i] / Ro0_1[i] * PI*d*d*(Ro0_1[i] / alpha[i])*Cd[i] * dt / 8;
                    if (n0[i] < 10e-50)
                    {
                        dP2[i] = 0;
                        df2[i] = 0;
                    }
```

```
                else
                {
                        dP2[i] = (PIntermediate[i] - PIntermediate[i - 1] + q1[i] -
q2[i])*dt / (dx*(Ro0_2[i] / (1 - alpha[i])));
                        df2[i] = -n0[i] / Ro0_2[i] * PI*d*d*(Ro0_1[i] /
alpha[i])*Cd[i] * dt / 8;
                }
                dV[i] = ((V[i] - U[i]) - (dP1[i] - dP2[i])) / (1 - (df2[i] - df1[i]));
        }

        dP1[0] = dP1[1];
        dP1[N - 1] = dP1[N - 2];
        dP2[0] = dP2[1];
        dP2[N - 1] = dP2[N - 2];
        df1[0] = df1[1];
        df1[N - 1] = df1[N - 2];
        df2[0] = df2[1];
        df2[N - 1] = df2[N - 2];
        dV[0] = dV[1];
        dV[N - 1] = dV[N - 2];

        //Euler part
        /*------------------------------------------------------------------------*/
        for (i = 1; i < N - 1; i++)
        {
                VIntermediate[i] = V[i] - dP1[i] - df1[i] * dV[i];
        }
        VIntermediate[0] = VIntermediate[1];
        VIntermediate[N - 1] = -VIntermediate[N - 2];
        for (i = 1; i < N - 1; i++)
        {
                UIntermediate[i] = U[i] - dP2[i] - df2[i] * dV[i];
        }
        UIntermediate[0] = UIntermediate[1];
        UIntermediate[N - 1] = UIntermediate[N - 2];
```

```
for (i = 1; i < N - 1; i++)
{
	if (n0[i] < 10e-50)
	{
		EIntermediate_2[i] = 0;
	}
	else
	{
		EIntermediate_2[i] = E0_2[i] + (UIntermediate[i] * UIntermediate[i] / 2 - U[i] * U[i] / 2);
	}
}
EIntermediate_2[0] = EIntermediate_2[1];
EIntermediate_2[N - 1] = EIntermediate_2[N - 2];

for (i = 1; i < N - 1; i++)
{
	alpha105 = (alpha[i] + alpha[i + 1]) / 2.0;
	alpha205 = ((1 - alpha[i]) + (1 - alpha[i])) / 2.0;
	EIntermediate_1[i] = E0_1[i] + ((E0_2[i] - EIntermediate_2[i])*Ro0_2[i] - ((PIntermediate[i] + q1[i]) * (alpha105*(VIntermediate[i] + VIntermediate[i + 1]) / 2.0 + alpha205*(UIntermediate[i] + UIntermediate[i + 1]) / 2.0) - (PIntermediate[i - 1] + q2[i]) * (alpha105*(VIntermediate[i] + VIntermediate[i - 1]) / 2.0 + alpha205*(UIntermediate[i] + UIntermediate[i - 1]) / 2.0))*(dt / dx)) / Ro0_1[i];
	if (EIntermediate_1[i] < 0)
		cout << i*dx << "\tE = " << EIntermediate_1[i] << endl;
}
EIntermediate_1[0] = EIntermediate_1[1];
EIntermediate_1[N - 1] = EIntermediate_1[N - 2];

/*---------------------------------------------------------------------------*/

//Lagrange part
/*---------------------------------------------------------------------------*/
```

```
//По Белоцерковскому
//1st phase mass translation calc
/*-------------------------------------------------------------------------*/
MassTranslation(RightBorderMassFlow1[1], LeftBorderMassFlow1[1], VIntermediate[1], Ro0_1[1], VIntermediate[1 - 1], VIntermediate[1 + 1], Ro0_1[1 - 1], Ro0_1[1 + 1], dt, dx, true, true);
LeftBorderMassFlow1[0] = LeftBorderMassFlow1[1]; RightBorderMassFlow1[0] = RightBorderMassFlow1[1];
for (i = 2; i < N - 2; i++)
{
    MassTranslation(RightBorderMassFlow1[i], LeftBorderMassFlow1[i], VIntermediate[i], Ro0_1[i], VIntermediate[i - 1], VIntermediate[i + 1], Ro0_1[i - 1], Ro0_1[i + 1], dt, dx, true, true);
}

MassTranslation(RightBorderMassFlow1[N - 2], LeftBorderMassFlow1[N - 2], VIntermediate[N - 2], Ro0_1[N - 2], VIntermediate[N - 2 - 1], VIntermediate[N - 2 + 1], Ro0_1[N - 2 - 1], Ro0_1[N - 2 + 1], dt, dx, true, true);
LeftBorderMassFlow1[N - 1] = RightBorderMassFlow1[N - 1] = 0;// LeftBorderMassFlow1[N - 2];
/*-------------------------------------------------------------------------*/

//2nd phase mass translation calc
/*-------------------------------------------------------------------------*/
MassTranslation(RightBorderMassFlow2[1], LeftBorderMassFlow2[1], UIntermediate[1], Ro0_2[1], UIntermediate[1 - 1], UIntermediate[1 + 1], Ro0_2[1 - 1], Ro0_2[1 + 1], dt, dx, true, true);
LeftBorderMassFlow2[0] = LeftBorderMassFlow2[1];
RightBorderMassFlow2[0] = RightBorderMassFlow2[1];//
for (i = 2; i < N - 2; i++)
{
    MassTranslation(RightBorderMassFlow2[i], LeftBorderMassFlow2[i], UIntermediate[i], Ro0_2[i], UIntermediate[i - 1], UIntermediate[i + 1], Ro0_2[i - 1], Ro0_2[i + 1], dt, dx, true, true);
```

```
}

MassTranslation(RightBorderMassFlow2[N - 2], LeftBorderMassFlow2[N - 2], UIntermediate[N - 2], Ro0_2[N - 2], UIntermediate[N - 2 - 1], UIntermediate[N - 2 + 1], Ro0_2[N - 2 - 1], Ro0_2[N - 2 + 1], dt, dx, true, true);
LeftBorderMassFlow2[N - 1] = LeftBorderMassFlow2[N - 2];
RightBorderMassFlow2[N - 1] = RightBorderMassFlow2[N - 2];
/*-------------------------------------------------------------------------*/

//Particle number translation calc
/*-------------------------------------------------------------------------*/
MassTranslation(RightBorderParticleFlow[1], LeftBorderParticleFlow[1], UIntermediate[1], n0[1], UIntermediate[1 - 1], UIntermediate[1 + 1], n0[1 - 1], n0[1 + 1], dt, dx, true, true);
LeftBorderParticleFlow[0] = LeftBorderParticleFlow[1]; RightBorderParticleFlow[0] = RightBorderParticleFlow[1];
for (i = 2; i < N - 2; i++)
{
    MassTranslation(RightBorderParticleFlow[i], LeftBorderParticleFlow[i], UIntermediate[i], n0[i], UIntermediate[i - 1], UIntermediate[i + 1], n0[i - 1], n0[i + 1], dt, dx, true, true);
}

MassTranslation(RightBorderParticleFlow[N - 2], LeftBorderParticleFlow[N - 2], UIntermediate[N - 2], n0[N - 2], UIntermediate[N - 2 - 1], UIntermediate[N - 2 + 1], n0[N - 2 - 1], n0[N - 2 + 1], dt, dx, true, true);
LeftBorderParticleFlow[N - 1] = LeftBorderParticleFlow[N - 2];
RightBorderParticleFlow[N - 1] = RightBorderParticleFlow[N - 2];
/*-------------------------------------------------------------------------*/

//Final part
/*-------------------------------------------------------------------------*/
//По Белоцерковскому
for (i = 0; i < N; i++)
{
```

```
                Ro1_1[i] = Ro0_1[i] + (LeftBorderMassFlow1[i] - RightBorderMassFlow1[i]) / dx;
                Ro1_2[i] = Ro0_2[i] + (LeftBorderMassFlow2[i] - RightBorderMassFlow2[i]) / dx;
                n1[i] = n0[i] + (LeftBorderParticleFlow[i] - RightBorderParticleFlow[i]) / dx;
                alpha[i] = (1 - n1[i] * PI*d*d*d / 6);
        }

        for (i = 1; i < N - 1; i++)
        {
                if (n1[i] <= 10e-50)
                {
                        E1_2[i] = 0;
                        U1[i] = 0;
                }
                else
                {
                        E1_2[i] = Ro0_2[i] / Ro1_2[i] * EIntermediate_2[i] + (1.0 / (Ro1_2[i] * dx))*(flowDirection(i - 1, UIntermediate, EIntermediate_2) * LeftBorderMassFlow2[i] - flowDirection(i, UIntermediate, EIntermediate_2) * RightBorderMassFlow2[i]);
                        U1[i] = Ro0_2[i] / Ro1_2[i] * UIntermediate[i] + (1.0 / (Ro1_2[i] * dx))*(flowDirection(i - 1, UIntermediate, UIntermediate) * LeftBorderMassFlow2[i] - flowDirection(i, UIntermediate, UIntermediate) * RightBorderMassFlow2[i]);
                }
                E1_1[i] = Ro0_1[i] / Ro1_1[i] * EIntermediate_1[i] + (1.0 / (Ro1_1[i] * dx))*(flowDirection(i - 1, VIntermediate, EIntermediate_1) * LeftBorderMassFlow1[i] - flowDirection(i, VIntermediate, EIntermediate_1) * RightBorderMassFlow1[i]);
                V1[i] = Ro0_1[i] / Ro1_1[i] * VIntermediate[i] + (1.0 / (Ro1_1[i] * dx))*(flowDirection(i - 1, VIntermediate, VIntermediate) * LeftBorderMassFlow1[i] - flowDirection(i, VIntermediate, VIntermediate) * RightBorderMassFlow1[i]);
```

```
                    P1[i] = Ro1_1[i] * (E1_1[i] - V1[i] * V1[i] / 2.0) *
0.4;//R*(E1[i] / cv)*Ro[i];
                }
                /*---------------------------------------------------------------------------*/

                E1_1[0] = E1_1[1];
                E1_2[0] = E1_2[1];
                E1_1[N - 1] = E1_1[N - 2];
                E1_2[N - 1] = E1_2[N - 2];
                P1[0] = P1[1];
                P1[N - 1] = P1[N - 2];
                V1[0] = V1[1];
                V1[N - 1] = -V1[N - 2];
                U1[0] = U1[1];
                U1[N - 1] = U1[N - 2];
                Ro1_1[0] = Ro1_1[1];
                Ro1_1[N - 1] = Ro1_1[N - 2];
                Ro1_2[0] = Ro1_2[1];
                Ro1_2[N - 1] = Ro1_2[N - 2];
                //Array swap
                /*---------------------------------------------------------------------------*/
                temp = P;
                P = P1;
                P1 = temp;

                temp = V;
                V = V1;
                V1 = temp;

                temp = U;
                U = U1;
                U1 = temp;

                temp = E0_1;
                E0_1 = E1_1;
```

```
E1_1 = temp;

temp = E0_2;
E0_2 = E1_2;
E1_2 = temp;

temp = e0_2;
e0_2 = e1_2;
e1_2 = temp;

temp = Ro0_1;
Ro0_1 = Ro1_1;
Ro1_1 = temp;

temp = Ro0_2;
Ro0_2 = Ro1_2;
Ro1_2 = temp;

temp = n0;
n0 = n1;
n1 = temp;

for (i = 0; i < N; i++)
	S[i] = P[i] / pow(Ro0_1[i], gamma);
/*---------------------------------------------------------------------------*/
t += dt;
if (loopCount % 100 == 0)
{
	ExcessPressure2[outputLoopCount] = P[N - 3] - p0;

	ParticleImpulse3[outputLoopCount] = Ro0_2[N - 4] * U[N - 4]
* U[N - 4] / 2;

	outputLoopCount++;
}
if (ExcessPressure2[outputLoopCount - 1] > maxEPressure)
```

```
                maxEPressure = ExcessPressure2[outputLoopCount-1];
            if (ParticleImpulse3[outputLoopCount - 1] > maxPImpulse)
                maxPImpulse = ParticleImpulse3[outputLoopCount - 1];
            if (loopCount % 1000 == 0)
                //cout << t << "\t" << ExcessPressure1[loopCount] << "\t" << ExcessPressure2[loopCount] << "\t" << ExcessPressure3[loopCount] << "\t" << ExcessPressure4[loopCount] << endl;
                //cout << t << "\t" << ParticleImpulse1[loopCount] << "\t" << ParticleImpulse2[loopCount] << "\t" << ParticleImpulse3[loopCount] << "\t" << ParticleImpulse4[loopCount] << endl;
                cout << t << "\t" << ExcessPressure2[outputLoopCount - 1] << "\t" << ParticleImpulse3[outputLoopCount - 1] << endl;

            loopCount++;
            //if (loopCount == int(0.0009 / dt) || loopCount == int(0.0028 / dt) || loopCount == int(0.0046 / dt) || loopCount == int(0.0055 / dt) || loopCount == int(0.0058 / dt) || loopCount == int(0.0069 / dt) || loopCount == int(0.0073 / dt) || loopCount == int(0.0083 / dt) || loopCount == int(0.0148 / dt) )
            //{
            ////output(Co, N, t, dx, V, "Vn");
            ////output(Co, N, t, dx, P, "Pn");
            ////output(Co, N, t, dx, Ro, "Ron");
            ////output(Co, N, t, dx, E, "En");
            //      //output(N, tstop, alpha1, d, dx, P, "P");
            //}
        }

        //File output
        /*-------------------------------------------------------------------------*/
        output(outputLoopCount-1, tstop, alpha1, d, dt, ExcessPressure2, "ExcessPressure");
        output(outputLoopCount-1, tstop, alpha1, d, dt, ParticleImpulse3, "ParticleImpulse");
        /*-------------------------------------------------------------------------*/
        cout << maxEPressure << "\t" << maxPImpulse << endl;
```

```
        system("PAUSE");
        DebugLog.close();
        return 0;
    }

    void output(int N, double tstop, double alpha, double d, double dx, double *fn, string
s)
    {
        int i;
        ofstream out;
        out.open(s + "N" + to_string(N) + "tstop" + to_string(tstop) + "alpha" +
to_string(alpha) + "d" + to_string(d) + ".dat");
        for (i = 0; i < N; i++)
        {
            out << fn[i] << endl;
        }
        out.close();
    }

    double flowDirection(int index, double* VIntermediate, double* value, bool
isFirstPhase)
    {
        double Uk05 = (VIntermediate[index] + VIntermediate[index + 1]) / 2;
        double Uk05l = VIntermediate[index] + (VIntermediate[index + 1] -
VIntermediate[index - 1]) / 4;
        double Uk05r = VIntermediate[index + 1] - (VIntermediate[index + 2] -
VIntermediate[index]) / 4;
        if (isFirstPhase)
        {
            if (Uk05 > 0)
            {
                if (VIntermediate[index] - VIntermediate[index + 1] >= 0)
                    return value[index] * Uk05;
                else
                {
```

```
                        if (Uk05l > 0)
                            return (value[index] + (value[index + 1] -
value[index - 1]) / 4)*Uk05l;
                        else
                            return 0;
                    }
                }
                else
                {
                    if (VIntermediate[index] - VIntermediate[index + 1] >= 0)
                        return value[index + 1] * Uk05;
                    else
                    {
                        if (Uk05r < 0)
                            return (value[index + 1] - (value[index + 2] -
value[index]) / 4)*Uk05r;
                        else
                            return 0;
                    }
                }
            }
            else
            {
                if (Uk05 >= 0 && Uk05l>0)
                    return (value[index] + (value[index + 1] - value[index - 1]) /
4)*Uk05l;
                else
                    return 0;

                if (Uk05 < 0 && Uk05r < 0)
                    return (value[index + 1] - (value[index + 2] - value[index]) /
4)*Uk05r;
                else
                    return 0;
            }
```

```
}

double flowDirection(int index, double* VIntermediate, double* value)
{
	if (VIntermediate[index - 1 + 1] + VIntermediate[index + 1] == 0)
		return 0;
	else
	{
		if (VIntermediate[index - 1 + 1] + VIntermediate[index + 1] > 0)
			return value[index];
		else
			return value[index + 1];
	}

}

//–асчитывает потоки массы через левую и правую границы ¤чейки.
void MassTranslation(double& RightBorderdM, double& LeftBorderdM, double V1, double Ro1, double leftCellVelocity1,
		double rightCellVelocity1, double leftCellDensity1, double rightCellDensity1, double dt, double dx, bool leftBorderAble, bool rightBorderAble)
{
	double leftBorderTranslationVelocity;
	double rightBorderTranslationVelocity;

#if DEBUG==1&&DETAILED==1&&SPEC==1
	cout << " V1 = " << V1 << " Ro1 = " << Ro1 << " leftCellDensity1 = " << leftCellDensity1 << " rightCellDensity1 = " << rightCellDensity1 << endl;
#endif
	//Right border processing
	if (rightBorderAble)
	{
		if (V1 + rightCellVelocity1 > 0)
		{
```

```
                        RightBorderdM = Ro1*(V1 + rightCellVelocity1)*dt / 2.0;

                }
                else
                {
                        if (V1 + rightCellVelocity1 < 0)
                                RightBorderdM = rightCellDensity1*(V1 + rightCellVelocity1)*dt / 2.0;
                        else
                                RightBorderdM = 0;
                }
        #if DEBUG==1&&DETAILED==1&&SPEC==1
                cout << "RightBorderdM = " << RightBorderdM << endl;
        #endif
        }
        else
                RightBorderdM = 0;
        //Left border processing
        if (leftBorderAble)
        {
                if (leftCellVelocity1 + V1 > 0)
                {
                        LeftBorderdM = leftCellDensity1*(leftCellVelocity1 + V1)*dt / 2.0;
                }
                else
                {
                        if (leftCellVelocity1 + V1 < 0)
                                LeftBorderdM = Ro1*(leftCellVelocity1 + V1)*dt / 2.0;
                        else
                                LeftBorderdM = 0;
                }
        #if DEBUG==1&&DETAILED==1&&SPEC==1
                cout << "LeftBorderdM = " << LeftBorderdM << endl;
        #endif
```

```
}
else
        LeftBorderdM = 0;}
```